

# **Exploration for autonomous 3D voxel mapping of static indoor environments with depth cameras and 2D odometry**

Thomas Wohlfahrt



## MASTERARBEIT

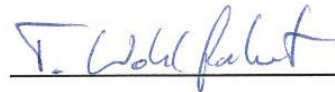
# EXPLORATION FOR AUTONOMOUS 3D VOXEL MAPPING OF STATIC INDOOR ENVIRONMENTS WITH DEPTH CAMERAS AND 2D ODOMETRY

Freigabe:

Der Bearbeiter:

Unterschriften

Thomas Wohlfahrt

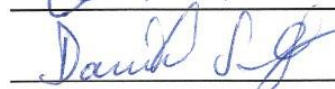


Betreuer:

Christian Rink



Daniel Seth



Der Institutsdirektor:

Dr. Alin Albu-Schäffer



Dieser Bericht enthält ~~82~~Seiten, ~~35~~ Abbildungen und ~~2~~ Tabellen



Technische Universität München  
Institute for Media Technology  
Prof. Dr.-Ing. Eckehard Steinbach



Deutsches Zentrum für Luft- und  
Raumfahrt e. V.  
Institute of Robotics and Mechatronics

# Master-Arbeit

Exploration for autonomous 3D voxel mapping of  
static indoor environments with depth cameras and  
2D odometry

Author:	Thomas Wohlfahrt
Matriculation Number:	03611145
Address:	Steinickeweg 7 80798 München
Email:	<code>thomas.wohlfahrt@tum.de</code>
Supervising Professor:	Prof. Dr.-Ing. Eckehard Steinbach
Supervisors (DLR):	Dipl.-Math. Christian Rink Dipl.-Math. techn. Daniel Seth
Begin:	03.11.2014
End:	01.05.2015

# Abstract

This thesis focuses on the development of an autonomous exploration and 3D mapping algorithm for a mobile service platform in an indoor environment. The designated robot platform is the KUKA omniRob additionally equipped with eight Time of Flight (ToF) cameras and the algorithm is developed and tested within the institute's internal Mobile Robot Environment (MRE) simulator. The Simultaneous Localization and Mapping (SLAM) problem is solved by a metaview registration method, in which range measurements and their corresponding odometry estimate are combined together by the Iterative Closest Point (ICP) algorithm. The algorithm operates on the raw range measurement without any feature generation. The result of the mapping process is a 3D occupancy grid map used for the autonomous exploration tasks. An information gain and frontier-based exploration routine is developed, based on a preceding discussion and analysis of literature on the subject. The implemented strategy uses an iterative approach for exploring the complete working space of the robot. Therefore, a collision free, traversable and reachable space is defined. This thesis introduces a 2D exploration grid map as a projection of the 3D occupancy grid map, which serves as an input for the given path planning process. The experiments within the MRE simulation environment demonstrate the applicability of the developed algorithm and give a critical analysis of existing assets and drawbacks. An autonomously working exploration algorithm is developed in this thesis, which provides a framework for further extensions and improvements to facilitate future researches.

**Keywords:** mobile robot, 3D mapping, active simultaneous localization and mapping, SLAM, metaview registration, ICP, autonomous exploration, frontier-based exploration, next best view planning, multi-view ToF cameras

# Abbreviations

<b>2D</b>	Two-dimensional
<b>3D</b>	Three-dimensional
<b>BFS</b>	Breadth-First Search
<b>CML</b>	Concurrent Mapping and Localization
<b>DLR</b>	German Aerospace Center ( <i>"Deutsches Zentrum für Luft- und Raumfahrt"</i> )
<b>DoF</b>	Degrees of Freedom
<b>EKF</b>	Extended Kalman Filter
<b>FoV</b>	Field of View
<b>ICP</b>	Iterative Closest Point
<b>L3D</b>	Library Lib3D
<b>LED</b>	Light-Emitting Diode
<b>LSR</b>	Local Safe Region
<b>LIDAR</b>	Light Detection and Ranging
<b>MRE</b>	Mobile Robot Environment
<b>NBV</b>	Next Best View
<b>PMD</b>	Photonic Mixing Device
<b>RAM</b>	Random Access Memory
<b>RBPF</b>	Rao-Blackwellized Particle Filter
<b>SLAM</b>	Simultaneous Localization and Mapping
<b>SRT</b>	Sensor-Based Random Tree
<b>ToF</b>	Time of Flight

# Mathematical Notation

$x$	variable
$\mathbf{x}$	vector
$\mathbf{X}$	matrix
$1 : k$	$= 1, 2, \dots, k$
$\mathbf{X}_{1:k}$	set of matrices $\mathbf{X}_{1:k} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_k\}$
$\mathbf{R}$	rotation matrix
$\mathbf{T}$	homogeneous transformation matrix
${}^s\mathbf{x}$	cartesian point with respect to coordinate system $s$
$\mathbf{r}$	robot pose vector $\mathbf{r} = (x, y, \theta)^T$
${}^w\mathbf{T}_r$	transformation matrix from $r$ - to $w$ -coordinate system (e.g. ${}^w\mathbf{x} = {}^w\mathbf{T}_r {}^r\mathbf{x}$ )
$th$	threshold value
$l_{res}$	grid resolution

# Contents

<b>Contents</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Description . . . . .	2
1.3 Thesis Structure . . . . .	4
<b>2 Related Work</b>	<b>5</b>
2.1 Kalman Filter . . . . .	5
2.2 Particle Filter . . . . .	6
2.3 Graph-based Optimization . . . . .	7
2.4 Robot Exploration . . . . .	8
2.5 Frontier-based Exploration . . . . .	9
2.6 Information Gain-based Exploration . . . . .	10
2.7 Polygon-based Map Representation . . . . .	11
2.8 Greedy Mapping . . . . .	11
2.9 Sensor-Based Random Tree (SRT) Method . . . . .	12
2.10 Next Best View Planning . . . . .	13
<b>3 Fundamentals</b>	<b>15</b>
3.1 Probabilistic Formulation of SLAM . . . . .	15
3.2 Robot Odometry . . . . .	17
3.3 Range Sensors . . . . .	18
3.4 Multiview Range Image Registration . . . . .	20
3.5 Map Representation . . . . .	21
3.5.1 Point Maps . . . . .	23
3.5.2 Occupancy Grid Maps . . . . .	24
3.5.3 Collision Space in Grid Maps . . . . .	25
3.6 ICP Range Image Registration . . . . .	26
3.6.1 The Algorithm . . . . .	26
3.6.2 Overlapping and Noise . . . . .	27
3.7 Entropy-based Exploration . . . . .	27
<b>4 Autonomous Exploration Process</b>	<b>30</b>
4.1 Mobile Robot Environment . . . . .	30

4.1.1	OmniRob Platform . . . . .	30
4.1.2	O3D100 Photonic Mixing Device . . . . .	31
4.1.3	L3D C++ Library . . . . .	33
4.1.4	Simulation Environment . . . . .	33
4.2	System Overview . . . . .	34
4.3	Map Notation . . . . .	35
4.4	SlamICP Process . . . . .	35
4.4.1	Robot Pose Estimation . . . . .	36
4.4.2	Pose Refinement . . . . .	37
4.4.3	Occupancy Grid Mapping . . . . .	39
4.5	Exploration Process . . . . .	40
4.5.1	Projection to Ground . . . . .	42
4.5.2	Collision Space Generation . . . . .	44
4.5.3	Frontier Generation . . . . .	45
4.5.4	Application of Exploration Strategy . . . . .	46
4.5.5	Path Generation . . . . .	48
4.6	Motion Control . . . . .	49
<b>5</b>	<b>Experiments and Discussion</b>	<b>50</b>
5.1	Example Exploration Process . . . . .	50
5.2	Exploration Performance . . . . .	54
5.2.1	Point Cloud Reduction . . . . .	56
5.2.2	Grid Resolution . . . . .	57
5.3	Major Drawbacks . . . . .	58
5.3.1	Metascan Misalignment . . . . .	58
5.3.2	Missing Map Segmentation . . . . .	61
5.3.3	Noisy Measurement . . . . .	62
<b>6</b>	<b>Conclusion</b>	<b>63</b>
6.1	Summary . . . . .	63
6.2	Future Work . . . . .	64
	<b>List of Figures</b>	<b>66</b>
	<b>List of Tables</b>	<b>67</b>
	<b>Bibliography</b>	<b>68</b>



# Chapter 1

## Introduction

### 1.1 Motivation

For the last few decades, robots have increasingly infiltrated our everyday life. They are used in assembly lines, as vacuum cleaners, for assisting surgeries or in many other fields, since highly involved algorithms make this development possible.

With robots, it is possible to enhance and expand the physical abilities of humans or to reduce the risk of human casualties in dangerous operations. Currently, autonomous acting of robots in unknown environments is of very high interest, both in the scientific and the economical aspect. The *Google Self-Driving Car* project exemplifies the importance and the attention spent towards this field of research. Furthermore, the number of robot-assisted surgeries grew from 80 000 in 2007 to 205 000 in 2011 and is still growing [BG10].

Another research topic investigates the ability of the human brain to represent the environment and how this can be adapted towards mobile robotics. In 2014, the *Nobel Prize in Physiology or Medicine* was awarded for "*Discoveries of cells that constitute a positioning system in the brain*".<sup>1</sup> This is a step towards a more intricate understanding of the human internal mapping process and might be useful for further development of mobile robots. However, in unknown environments and with unsupervised operations, robots are still very limited in their capabilities. Detecting dynamic obstacles and responding to unpredicted events seem very basic tasks for human beings, but the development of algorithms for these tasks is particularly complex and elaborate. Failures and misbehaviors of a robot might lead to serious damage or even cost human lives and are therefore not acceptable. The creation of reliable algorithms for autonomously acting robots is a very multi-layered and sophisticated task.

---

<sup>1</sup>[www.nobelprize.org/nobel\\_prizes/medicine/laureates/2014/](http://www.nobelprize.org/nobel_prizes/medicine/laureates/2014/) (Link: 25.03.2015)

The necessary tasks for a robot to act autonomously are:

1. Perceive the environment
2. Interpret the perceived data
3. Deduce a reasonable action to be performed

For the perception, appropriate sensors are needed, whereas for the interpretation, a representation model has to be found on which further processing can be based. For many navigation and localization algorithms, the processing of any interaction is based on an appropriate map representation. Building its own map representation is essential for an autonomous behavior of a robot and will be discussed in this thesis.

At the institute, where this thesis is performed, the map building process for the relevant robot is yet done manually by taking measurements of the real environment and then modeling it with 3D computer graphics software like Blender.<sup>2</sup> The designated goal of the present work is the execution of this task by the robot without any human interaction. When the robot is switched on in a new environment, it should be able to drive around in a well-defined manner to gather data and build an internal 3D map representation. This map can serve as a basis for more involved and dynamically reactive algorithms and for application in real world scenarios. The whole process of an autonomous environment building can be split into three interconnected tasks: *mapping*, *localization* and *path planning*. They are the focus in this thesis and a solution for a combined application of these tasks is presented in the following.

## 1.2 Problem Description

In [TMM<sup>+</sup>06] the problem is defined in the following way: "*Autonomous robots must possess the ability to explore their environments, build representations of those environments, and then use those representations to navigate effectively in those environments.*" This statement corresponds to the underlying problem in this thesis and can be split into two parts:

1. **Simultaneous Localization and Mapping:** The process of building a map of the environment while simultaneously localizing the robot in it. With the help of sensor data, an appropriate mapping process and representation has to be found. In literature this problem is abbreviated with **SLAM** and refers to the task of making the noisy perception data manageable.
2. **Exploration:** Exploration defines the task of autonomous and active gathering of information to acquire an accurate and complete model of the envi-

---

<sup>2</sup><http://www.blender.org/> (Link: 25.03.2015)

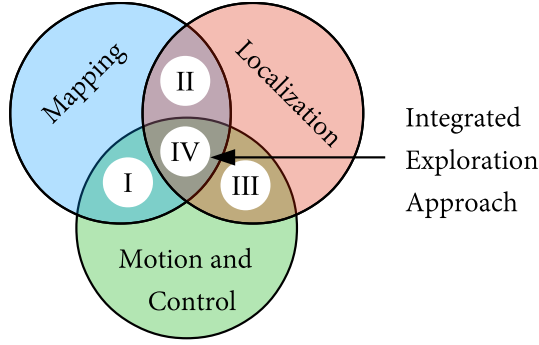


Figure 1.1: The different parts of robotic exploration [MWBDW02]: (I) classic exploration, (II) SLAM, (III) active localization, (IV) integrated exploration.

ronment. The selected SLAM algorithm will be extended by the task of path planning and following, in order to build a complete map.

The combination of these tasks can be considered an integrated approach, as can be seen in Figure 1.1. These tasks have all been well studied in literature and the following different sections are defined [MWBDW02]:

- **Active Localization** (Figure 1.1 III) comprises the task of finding control actions to maximize the probability of a correct localization in an already known map.
- **SLAM** (Figure 1.1 II) denotes the problem of mapping an environment and localizing in it at the same time, as mentioned previously.
- **Classical Exploration** (Figure 1.1 I) refers solely on planning where to go next, in order to get a complete environment model. Uncertainties in localization and mapping are not considered explicitly.
- **Integrated Exploration** (Figure 1.1 IV) is a holistic approach, which combines the previous tasks. The exploration task also fully includes mapping and localization. The process of generating the next targeted exploration position and path following also has to consider map inaccuracies and localization errors. Therefore, a strategy has to be found for selecting a path, where a high information gain can be expected. Additionally, this strategy should maximize the probability of a correct localization as well.

All problems in Figure 1.1 are all well known and studied in literature, but for the most part considered independently. This thesis wants to give a holistic outline and solution proposal for the SLAM problem with an autonomous exploration strategy. The main purpose of this work is to provide a working integrated solution for the KUKA omniRob mobile robot platform to autonomously create its own map representation. The institute's internal L3D library already provides algorithms for localization and path planning, but a model of the environment is needed in order to use them. For the acquisition of environment data, the eight ToF cameras at-

tached to the sides of the top-cover of the robot are used. Furthermore, the robot has an odometry sensor integrated, which gives an initial pose estimate about its current location relative to its starting location. The use case is limited to indoor environments and all movements will only be performed on a planar floor. Hence, no elevations, slants or stairways are regarded in this thesis. These considerations are left for future research work.

First, initial assumptions have to be made. The cameras are considered completely calibrated and their positioning towards each other and the robot is known. Additionally, the space around the initial starting position of the robot must be obstacle free within an adequate range. The region close to the robot cannot be perceived in the used camera positioning setting. Therefore, the first movement is always through unknown territory.

The mapping process requires the interaction of different tasks and processes. The first step is the acquisition of sensor data and the creation of an internal map representation. Thereafter, the map has to be analyzed and a strategy has to be found in order to determine a next best position to increase the map coverage. The resulting model is supposed to be accurate and consistent and the exploration strategies should be feasible within a limited magnitude of time and robot movements. Methods to cope with the sensor noise, have to be found and integrated. A simulation environment is available and serves as a testing tool. The thesis is completed within the time of six months.

## 1.3 Thesis Structure

In chapter 2, the relevant existing literature is analyzed and different solution approaches for the respective problem are presented. The most important theory topics, required to understand the underlying problem, are explained in chapter 3. The implemented algorithm for solution of the SLAM and exploration problem is explained in chapter 4. Additionally, the development environment with and used tools are mentioned. The implemented algorithm is further demonstrated and discussed in chapter 5, where a complete exploration process is demonstrated and the individual performance aspects are analyzed. Finally, the results and conclusions of this thesis are summarized in chapter 6. This chapter also presents possible future works and enhancements.

# Chapter 2

## Related Work

In this chapter, an overview in robotic mapping is given and some important developments in this field are presented. As stated by [MWBDW02], the integrated exploration approach is divided into the interconnected parts *localization*, *mapping* and *motion control*. The first two are related to as the SLAM problem and many different proposal solutions have been presented in the last two decades. This problem has often been referred to as a *solved problem*, but as soon as the aspects large-scale mapping, processing time and robustness are considered as criteria, very different and involved approaches have been developed. Each approach focuses on optimizing on of the mentioned aspects.

The first task in this thesis is to find an appropriate SLAM approach in order to generate a map representation of the environment by the interpretation of sensor data and locate the robot in it. The second task is the integration of the SLAM approach into an appropriate exploration routine, in order to fulfill the designated task of an autonomous and robust map building process. In the following, important approaches are pointed out and evaluated for the application in this thesis.

### 2.1 Kalman Filter

A first proposal solution to the SLAM problem in mobile robotics was introduced in 1986 by Hugh Durrant-Whyte [DW88] and Randall Smith et al. [SC86]. They presented estimation techniques to draw relationships between landmark locations and robot poses by using the Extended Kalman Filter (EKF) [Kal60]. Furthermore, they proposed methods to manage geometrical uncertainties with successive observations. A *landmark* can be seen as a specific or significant point in the environment detectable by a sensor.

Smith et al. [SSC90] further refined their idea and stated a foundation of feature-based mapping with point landmarks and *known data association*. This was the

birth of a powerful statistical framework to solve the simultaneous mapping and localization problem. From this point on, the abbreviations SLAM (*Simultaneous Localization and Mapping*) or CML (*Concurrent Mapping and Localization*) are used for the robotic mapping problem [T<sup>+</sup>02]. Moreover, different approaches and strategies for solving the SLAM problem were published. Leonardo et al. [LDW91] used extracted line segments from sonar sensor data, Gonzales et al. [GOR94] used laser data instead and Ayache et al. [AF88] even built visual 3D maps from a stereo camera system.

The convergence property had been an unsolved problem so far and the large computational complexity, which, for the EKF grows quadratically to the number of landmarks in the state vector, constrained the further development to a temporary halt. The focus has been on either localization or mapping as isolated problems [DWB06]. In 1995, Durrant-Whyte et al. [DWRN96] presented a convergence proof and clarified the structure of the SLAM problem. The break-through was the consideration of the mapping and localization problem as one interleaved estimation process. The EKF algorithm is still widely used in contemporary applications. Still, the quadratic growing complexity, proportional to the number of landmarks, is a major shortcoming. One way to cope with this effect is to divide the map into individual sub-maps or to apply approximations [GN01].

In this thesis, the raw camera data is used for the mapping and exploration process and no features or landmarks are extracted. The EKF assumes a known data association for every iteration step, whereas this is not achievable for the operation on the raw depth measurement. A robust feature extraction and data association is necessary for the deployment of an EKF method, therefore it is not suited as a solution to the SLAM problem in this thesis.

## 2.2 Particle Filter

Another successful approach to solve the SLAM problem is the Rao-Blackwellized Particle Filter (RBPF) [DDFMR00]. In order to represent probability distributions, sampling is used, and each particle represents its own path through the environment. Montemerlo and Thrun [MTKW02] used this theory to develop and implement the FastSLAM algorithm. With an efficient tree structure, for the occupancy grid map representation, the update step of the SLAM posterior could be performed in  $O(P \cdot \log(N))$  time complexity, where  $P$  is the number of particles and  $N$  the number of landmarks. In contrast to the  $O(N^2)$  update time of the EKF algorithm, considerably more landmarks could be handled. To decrease the number of required particles, FastSLAM 2.0 [MTKW03] uses the most recent measurement in order to create a more exact estimated pose distribution. The process of sampling the distribution is called *importance sampling*. It enables the effective generation of samples for an arbitrary probability distribution. The FastSLAM algorithm provides

a powerful way to solve the correspondence problem on an easy way because each particle represents its own position path and map. Non-linearities in robot motion models do not have to be linearized like in the EKF algorithm [MTS07].

The particle filter also enables the use of raw sensor data to build grid-maps of the environment in a robust manner [GSB07]. For the integrated exploration approach combining mapping, localization and exploration, Stachniss et al. present the use of the RBPF [BSG05]. Goal of the whole exploration process is to minimize the entropy of the robot trajectory and occupancy grid map for each particle. Each particle represents the environment for a specific trajectory estimate. The exploration strategy focuses on a trade-off between information gain and the active closing of loops in the environment, in order to reduce the localization error. The traveling costs are proportional to the state certainty of all traversed grid cells. A loop occurs if a robot returns to a past location, after having discovered new terrain for a while.

For the 3D case, the use of a particle filter has some considerable drawbacks. Each particle has to represent the world with its own 3D map and therefore, a lot of processing power and Random Access Memory (RAM) is needed. Jochen Welle et al. solved the problem by introducing a specific data structure called *DeltaOctree*, which describes the evolution of the particles by a specific tree structure [WSBC10]. The particles with a common history share map parts, and the amount of necessary particles is kept to a minimum.

This approach states an attractive solution for the SLAM problem in this thesis. But considering the missing availability of the DeltaOctree structure for the realization in software and the high implementation effort for combining this approach with an appropriate exploration strategy, this method is not in the scope of this thesis. In order to make the mapping and exploration process processable on a regular PC, only one single 3D occupancy grid map is used in this thesis as an environment representation. The focus is on the realization of a complete autonomous exploration process within the cope of six month and the deployment of an applicable and testable framework for future work. Therefore, only one memory saving 3D octree data structure is applied for the map representation and only one single pose estimate is performed.

## 2.3 Graph-based Optimization

Lu et. Milios [LM97] presented a different approach to SLAM. In order to solve the Full-SLAM problem, they constructed a sparse graph of *soft constraints*, consisting of observed corresponding landmarks and the different robot poses. In this information-theoretic approach, distinct range scans are aligned and by global non-linear optimization, a very accurate consistent map could be built. This stands in contrast to the filter-based approaches of the EKF or particle filter SLAM strate-

gies. Drawbacks are the increasing complexity proportional to the amount of measurements. Later publications referred to this approach as *Graph-based SLAM* or *Graph-SLAM* [GKSB10, TBF05]. This method is very common in current applications, but mostly for offline optimization. A further extension of this algorithm is the *sparse extended information filter* (SEIF) [TLK<sup>+</sup>04], in which the update steps can be performed in constant time regardless of the number of gathered features.

Andreas Nüchter describes an approach for a globally consistent range image registration for 3D data with 6 Degrees of Freedom (DoF) [Nüc09]. In the first step, the ICP algorithm applied to the raw sensor data is used to merge the scans into one common coordinate system. This process, called *registration*, is erroneous due to the accumulation of small alignment errors. These errors surface in case loops are detected. Therefore, in the second step, a graph is build out of the robot's pose relations. The accumulated error is then distributed over the loops by *global relaxation* in order to acquire a consistent map representation.

The SLAM system from Henry et al. [HKH<sup>+</sup>10] for RGB-D cameras like the Microsoft Kinect is a recent graph-based approach. The graph building process is supported by feature matching methods for the RGB images like SIFT or SURF and is called *Frontend*. The loops are closed and relaxed in the *Backend* by a global pose graph optimization algorithm like g2o [GSKB11]. This approach combines the visual features with a depth information-based shape alignment.

The cameras used in this thesis can only perceive depth data and no RGB images. Furthermore, no feature extraction process is performed. Therefore, the present cameras are not well applicable for the RGB-D SLAM strategy. The approach from Nüchter [Nüc09] is well suited for this thesis. It uses raw range data and works for the 3D case. As a drawback for the mapping process, every time a loop is detected, the whole graph can change. The 3D occupancy grid map, which is used as a basis for the exploration process, would have to be rebuild completely after each optimization step. Therefore, the SLAM approach in this thesis is based on 3D range scan alignment by ICP without the relaxation of the pose graph.

## 2.4 Robot Exploration

With the help of an appropriate SLAM algorithm, the robot is able use the acquired data from the odometry and depth image sensors to construct an internal map representation and to locate itself in it. But the sensors have a limited range and the whole map can normally not be perceived from one point of view. Moreover, there can be obstacles which are covered by other obstacles and hence cannot be detected by the sensors. In order to generate a holistic global map autonomously, the robot has to decide where to go next to perform a new measurement of the environment. One easy solution would be to drive the robot around in the environment guided by a



human. This solves the problem of obstacle avoidance and the decision making process of calculating a next best target pose, but also contradicts a fully autonomous behavior of the robot. To establish a self-governed robot, reasonable algorithms for these tasks have to be found.

Yamauchi B. defines exploration as *"the act of moving through an unknown environment while building a map that can be used for subsequent navigation"* [Yam97]. The act of deciding where to go next is referred to as *exploration strategy* and its goal is to *"generate a complete or nearly complete map in a reasonable amount of time"*.

This means, that not only the information gain of a new pose has to be considered, but also the cost of going there. Reasonable indicators for the cost are both, the necessary traveling time and also the probability of a correct localization of the robot. A *random walk* through the environment can therefore also be understood as an exploration strategy, but it is very likely not to be the optimal solution. As time goes by, this strategy might also cover the whole environment eventually, but in this thesis, a more target-based approach is presented.

The exploration task itself is related to well-known problems in literature like the illumination, the shortest watchmen or the art gallery problem [HBAB10]. The art gallery problem refers to the real-world problem of guarding an art gallery, with as little necessary guardians as possible. It has been proven by Cheval [Chv75], that a map, consisting of  $n$  polygons, can always be fully be monitored by  $\lfloor n/3 \rfloor$  watchmen. The problem of finding the minimum number of needed watchmen for an arbitrary map constellation is proved to be NP-complete and requires knowledge about the whole environment. Furthermore, it relates to tasks like inspection, tracking or surveillance of environments.

Many different approaches to the autonomous exploration problem have been proposed in the last two decades. Most of them are based on 2D maps [AG05, FO05, BATP10, SHB04], where the robot is only targeted for the use in a planar environment. Some line of research also focuses on the use of 3D maps for the exploration task [AC10, JSPB07, SNH03]. For integrated exploration approaches, the uncertainty in localization in their exploration process is considered as well [MWBDW02, TMM<sup>+</sup>06]. Whereas another research field deals with the strategic cooperation of multiple robots, in order to explore one interconnected environment [BMF<sup>+</sup>00, WSB08]. In this thesis, the robot is also limited to planar environments, but the exploration process is based on a 3D occupancy grid map representation.

## 2.5 Frontier-based Exploration

Yamauchi [Yam97] introduced the popular frontier-based exploration approach for grid maps. The state of each cell in the grid map is divided into either *free*, *occupied*

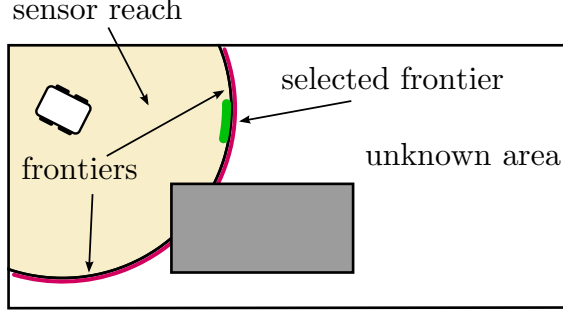


Figure 2.1: Frontier-based exploration. The separation between perceived and unknown environment is called frontier. After an evaluation process, a target position is selected.

or *unknown*, and it depends on the cell's probability of being occupied. In case the probability value of the cell equals the initial state, it is considered unexplored. The frontier cells are those free cells, which possess a neighboring unexplored cell. Connected frontier cells are combined to frontier regions, which, after surpassing a certain size, are interpreted as frontiers to unexplored area. Many of the proposed exploration strategies are more or less based on the generations of these frontiers. In Figure 2.1, the frontier-based exploration is visualized exemplarily. Out of the possible frontiers, a target position is chosen, based on a specified strategy. Often, sampling and arbitrary selection procedures are used in order to make the decision process assessable.

The exploration strategy in this thesis also follows the frontier-based approach, but the selection process of target positions is based on the data from a 3D occupancy grid map. The path planning and frontier extraction is also performed on a 2D grid map. Additional collision spaces and sampling areas are defined in the present proposal to integrate the consideration of the robot's geometry and to reduce the processing time. Furthermore, due to the special sensor assembly, this approach has to be adapted appropriately.

## 2.6 Information Gain-based Exploration

In [MSW01], Moorehead et al. present an approach, where the selection of target poses is based on several distinct information resources. The environment is divided into a discrete grid, where every cell is associated with an information vector  $\mathbf{g} \in [0, 1]^n$ . Every entry is defined by a simulated measurement and resembles a different source of information. The final expected information gain is the weighted sum of all entries in  $\mathbf{g}$ . A sequence of traversable cells, including the positions, where a measurement of the environment shall be performed, defines the path from the current to the target location. The specific *gain* of a path is defined by the

combination of various criteria. It includes the expected information gain at the measurement points and the time, which is needed in order to drive through the path, take the observations and perform the planning. Calculating the path with the maximum information gain can be considered infeasible and its complexity resembles that of the NP-hard *prize-collection salesmen problem*. This is the reason for proposing a sub-optimal but processable strategy, where only the next measurement point is considered and the gain for every reachable point is calculated. For the implementation of the information gain in this strategy, the entropy of the cells is used. This approach is similar to the one used in this thesis. But the calculation of information gain in this thesis is based on a simulated measurement with 3D range sensors. Robot poses are sampled and evaluated in terms of information gain.

## 2.7 Polygon-based Map Representation

The exploration strategy proposed by González-Baños und Latombe [GBL02] is also frontier-based, but uses a polygon-based map representation. They use different categories of edges, where the separation lines between known and unknown area are defined as *free* edges. Sampling is used to determine the next robot pose and every possible sample must be in direct line-of-sight with a free edge. In the evaluation process, the potential visibility gain is calculated by the amount of visible area outside the explored region. A trade-off between the path-length to the target position and its information gain determines the value of a possible pose candidate. If no free edge surpasses a certain threshold, which serves as a selection criteria, the exploration terminates. The polygon-based map representation allows an efficient way of representing a 2D environment, but a prior line segmentation algorithm has to be performed on the raw sensor data. In this thesis, the map is represented by a 3D occupancy grid and no feature extraction algorithms are applied. But the sampling process for selecting new target poses is based on the same principle. Furthermore, the absence of any possible pose candidates conforms to the termination criterion as well.

## 2.8 Greedy Mapping

In [KTH01], Tovey and Koenig propose a *greedy mapping* strategy, where the robot always targets the closest not yet visited location until the complete map is explored. The environment is represented by a graph, where the vertices resemble robot poses and the edges the connection between them. They prove the existence of an upper-bound of  $O(|V| \log |V|)$  edge movements as a worst case scenario, where  $|V|$  is the number of vertices in the graph. Furthermore, they show that even this easy approach is not exceptionally far away from an optimal exploration strategy. No long

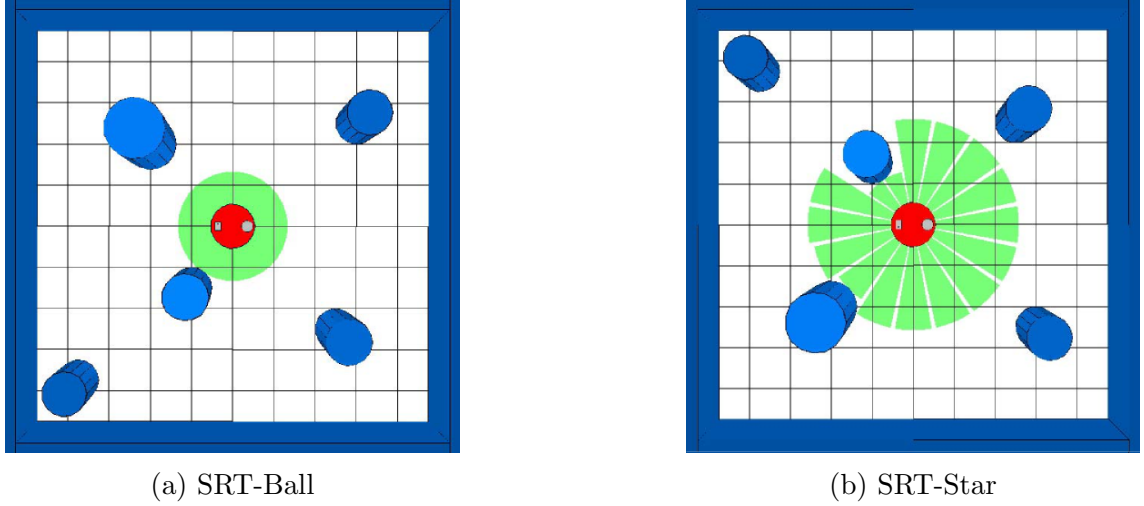


Figure 2.2: The SRT models from [OVFT04].

therm planning is performed, but for many cases, sufficient results can be achieved. The exploration proposal in this work also resembles a greedy approach. The distance to the selected frontier is the main criterion. But the exact pose in front of the frontier is generated by sampling and an information gain evaluation. This method is implemented to reduce the number of necessary environment recordings.

## 2.9 Sensor-Based Random Tree (SRT) Method

The basic principle of this method, proposed by Oriolo et al. [OVFT04], is to only drive the robot into regions it already has explored and which are in nearby sight, in order to avoid collisions. The target pose has to be within the current sensor reach of the robot and no intercepting collision must be present.

In the first step, the region around the robot is perceived, which is limited by the range of the sensors. For the 2D-case, these areas are modeled as a ball for the conservative, or as a star for the optimistic case and are depicted in Figure 2.2. In case of the ball, one obstacle already limits the range of the Local Safe Region (LSR), here depicted in green. At the star shaped model, the LSR is divided into several equally large cones. Only the cones in which the obstacle is present are limited in their range. All other cones remain within the sensor coverage. This process is performed analogously in case several obstacles are present. In this LSR, the robot can move freely without colliding. After the robot has traveled to a target position, it perceives its surroundings and integrates the new measurement in the Sensor-Based Random Tree (SRT) data structure. The data structure consists of the respective poses realized as nodes where the robot has already been, combined with the respective measurement data. The robot chooses his next observation point

based on the following criteria:

1. The next node has to be present in the current LSR of the robot.
2. It must not be located within the LSR of a previous node.
3. It has to keep a minimum distance to the current pose.

In case no such pose is found, the robot moves back to the previous node. This process is called *backtracking*. Therefore, several consecutive steps are performed. Initially, the exploration direction is chosen arbitrarily. The distance to drive is calculated by a product of the LSR radius and a specified constant between 0 and 1. This ensures the target position to be within the LSR of the current node. Now, the criteria points 2. and 3. are evaluated. The robot moves to this target point in case both terms are considered true. If they do not hold true, the generation process is started again by selecting a new exploration direction. The arbitrary selection of a target pose is very likely to deliver an appropriate goal after several tries [OVFT04]. In case a maximum number of sample points are evaluated as negative, this node is considered an end point, and the robot moves back to its previous pose. From this observation point, the robot starts the exploration process again. The SRT approach is performed by the random selection of target poses, based on the length of edge segments. In contrast, the approach in this thesis considers the estimated information gain for the target poses in combination with the traveling cost. The exact pose generation is also sampling based, but in the present proposal, no pose graph is composed and no backtracking performed. Instead, another frontier is searched in the whole environment.

## 2.10 Next Best View Planning

The necessary process steps for a Next Best View (NBV) strategy are described in Figure 2.3 according to [AG05]. While SLAM algorithms try to make an accurate map out of the sensory information, NBV algorithms want to find a new pose where the sensors can provide the best possible input, considering certain criteria. It is also a well known problem in the graphics and computer vision community [Pit96, BZW<sup>+</sup>95]. But these algorithms can not directly be applied for the purpose of mobile robot exploration.

First, the collision problem is approached in a different way. In object modeling, the sensor moves around relatively small objects in free space and does normally not have to consider collisions with it. Keeping a certain distance to the object, the sensor is able to move freely around the object. At an exploration execution by a mobile robot, the sensors are moved inside the to be mapped object and the task of avoiding collisions with the environment is of highest importance. Furthermore, the relocalization step has to be approached in a different manner. The odometry and

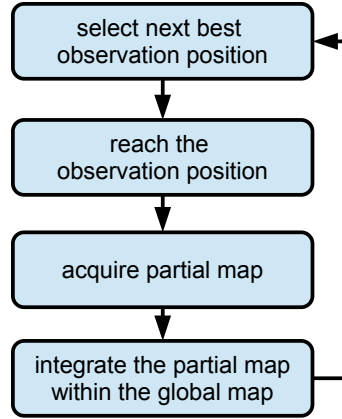


Figure 2.3: Next Best View exploration process as described in [AG05].

slippage errors of the robot have to be opposed and a constant localization relative to the partially constructed map has to be performed.

In NBV-planning for 3D object modeling, often sample points around the to be modeled object are generated, with the camera pointing towards the object. These sample points are evaluated, in terms of their *information gain*. If an octree data structure is used, there is a distinction between *free*, *unknown* and *occupied* space. The amount of *unknown* space, which can be perceived by such a sample point is an indication for the information gain of this configuration. This thesis uses the NBV approach for the exploration task. The generation of possible target poses is based on a 2D environment representation, whereas their evaluation is based on a 3D map representation. The external world relates to an indoor environment. The targeted decision output of the exploration process is a sequence of control actions in order to find the next best robot pose, based on the current internal map representation. The difficult part, which differs from the classical Next-Best View planning strategies in computer vision, is to guarantee a safe navigation with only partial knowledge about the environment. Enough overlap between every sensing pose has to be ensured as well. Otherwise, it might not be possible for the local map to be registered correctly into the global map representation.

# Chapter 3

## Fundamentals

The autonomous exploration process covers many distinct fields of research. In order to enable a better classification of the implemented algorithm, the important and fundamental topics for this thesis are presented in the following. In this chapter, no complete coverage of all relevant theory is given, but the most important aspects are mentioned. They form the basis for the implemented algorithm in chapter 4.

### 3.1 Probabilistic Formulation of SLAM

Most state-of-the-art algorithms are derived from either the EKF-based, Graph-based or Particle Filter-based SLAM approach [SK08, ch.37]. The tutorial papers [DWB06] and [BDW06] give a broad overview on the whole SLAM problem and its different approaches. In the following, the notation for formulating the SLAM problem is explained in more detail. At a specific time step  $t$  the following state, control and measurement vectors are defined:

- $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$ : represents the state vectors from time step  $t = 0$  to time step  $t = k$ . The time is discretized to describe the distinct and limited number of processing steps. At a specific time step  $t$  the vector  $\mathbf{x}_t$  represents the state of the robot, which is normally its position and orientation and  $\mathbf{x}_0$  describes the initial starting position. In the planar case,  $\mathbf{x} = (x, y, \theta)^T$  suffices to fully describe the state, whereas in the 3D case, the vector  $\mathbf{x} = (x, y, z, \theta_x, \theta_y, \theta_z)^T$  is necessary to describe the state fully. For every additional DoF, a new parameter is necessary.
- $\mathbf{U}_{1:k} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k\}$ : represents the control vectors up to the time step  $t = k$ .  $\mathbf{u}_t$  is the control vector in order to steer the robot from state  $\mathbf{x}_{t-1}$  to  $\mathbf{x}_t$ . It expresses the relative movement between the two robot poses and gives an initial estimation. By virtue of noise and odometry errors, these estimates

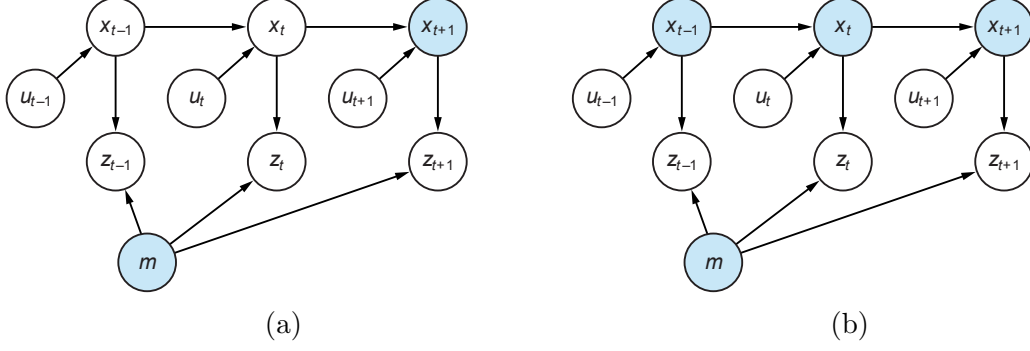


Figure 3.1: Online (a) and Full (b) SLAM Dynamic Bayesian Network (DBN) [GKSB10]. The DBN is a representation for the filter-based SLAM problem. The nodes in blue mark the targeted solution vectors.

can not complete be considered accurate. How the motion process is realized exactly is not relevant for the SLAM problem itself.

- $\mathbf{Z}_{1:k} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$ : represents the measurement vectors, which consist of the gathered sensor data at a distinct time step  $t$  and the state  $\mathbf{x}_t$ . A depth image, recorded by a ToF camera, would be an example of such a measurement.
- $\mathbf{m}$ : stands for the map of the sensed environment. This can be a set of landmarks of the environment or any other targeted representation.

The solution to the SLAM problem is the acquisition of an accurate model of the world  $\mathbf{m}$  and the robot locations  $\mathbf{X}_{0:k}$ . Many SLAM algorithms are probabilistic, in which a joint posterior probability distribution has to be calculated. In literature, there is a distinction between the *Online* and the *Full* SLAM problem. The joint posterior probability distribution for these problems is:

- $P(\mathbf{x}_k, \mathbf{m} | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k})$  for the *Online* SLAM and
- $P(\mathbf{X}_{1:k}, \mathbf{m} | \mathbf{U}_{1:k}, \mathbf{Z}_{1:k})$  for the *Full* SLAM problem.

This relation can be expressed graphically by a Dynamic Bayesian Network (BDN) in Figure 3.1. The control and measurement vectors are directly observable. Only the targeted hidden solution vectors, marked in blue, differ for either of the SLAM formulations. For the Online SLAM problem, only the current state vector  $\mathbf{x}_t$  needs to be estimated, whereas the Full SLAM problem targets the complete path  $\mathbf{X}_{1:t}$  starting from the beginning. The approach presented in this thesis currently only solves the Online SLAM problem. The location of the current robot pose is calculated and then the sensor data is used for the mapping task. For further extensions and developments, this approach could be extended to the solution of the Full SLAM problem, in case the graph-based formulation is pursued.



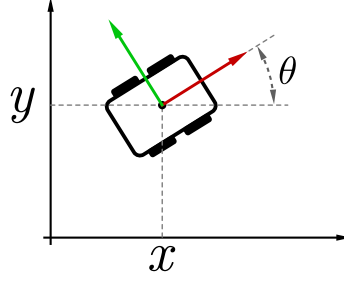


Figure 3.2: Robot pose in 2D coordinate frame. The pose can be fully described by the translation components  $x$  and  $y$ , and its rotation  $\theta$ .

## 3.2 Robot Odometry

Many robots have odometry sensors integrated, which can be used to get an estimate about the robot's current pose in a planar environment. The term *pose* defines the combination of the position and orientation of a rigid body in space. This pose is relative to the starting pose, where the robot has been switched on. The odometry counts or measures the wheel rotations and integrates the gathered data over the working time. But the longer the robot is driving around, the more this estimate is about to diverge from the robot's real pose in the environment. In Figure 3.2, the pose of the robot is depicted, which can be fully described by the vector  $(x, y, \theta)^T$  in the world reference frame  $w$ . The orientation  $\theta$  is often called *heading direction* or *bearing*. The robot has its own right-handed coordinate frame  $r$  from top-view, where the origin is located at its center and the  $x$ - and  $y$ -axes are marked in red and green color, respectively.

The notation for describing the transformations in this thesis is based on the *Springer Handbook of Robotics* [SK08]. To express the position of a point  ${}^r\mathbf{p} = (x, y)^T$  in robot reference frame  $r$  relative to the world reference frame  $w$ , the equation

$${}^w\mathbf{p} = {}^w\mathbf{R}_r {}^r\mathbf{p} + {}^w\mathbf{t}_r \quad (3.1)$$

is needed. The rotation matrix  ${}^w\mathbf{R}_r$  and the translation vector  ${}^w\mathbf{t}_r$  for the scenario in Figure 3.2 would be

$${}^w\mathbf{R}_r = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \text{ and } {}^w\mathbf{t}_r = \begin{pmatrix} x \\ y \end{pmatrix}, \text{ respectively.} \quad (3.2)$$

In order to combine the rotation matrix  ${}^w\mathbf{R}_r \in \mathbb{R}^{2 \times 2}$  and the translation vector  ${}^w\mathbf{t}_r \in \mathbb{R}^{2 \times 1}$ , the homogeneous transformation  ${}^w\mathbf{T}_r$  is used:

$$\begin{pmatrix} {}^w\mathbf{p} \\ 1 \end{pmatrix} = \begin{pmatrix} {}^w\mathbf{R}_r & {}^w\mathbf{t}_r \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} {}^r\mathbf{p} \\ 1 \end{pmatrix} = {}^w\mathbf{T}_r \begin{pmatrix} {}^r\mathbf{p} \\ 1 \end{pmatrix}. \quad (3.3)$$

The terms  $({}^w\mathbf{p} \ 1)^T$  and  $({}^r\mathbf{p} \ 1)^T$  represent the vectors  ${}^w\mathbf{p}$  and  ${}^r\mathbf{p}$  in homogeneous notation. For further information about homogeneous transformations, it is referred to [SK08, Ch. 1 Kinematics]. They enable the combination of a rotation

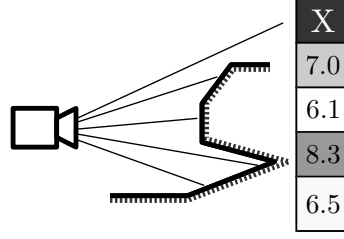


Figure 3.3: Working principle of range sensors. The values describe the measured distances of each individual pixel. The higher the distance, the darker is the pixel representation. In case it surpasses a maximum distance, no valid value can be returned.

and translation (Equation 3.1) into one single matrix multiplication (Equation 3.3). With a *rigid transformation*, all distances between pairs of points are preserved, therefore no scaling, shearing or squeezing is performed.

The pose of a robot on a plane relative to the world reference frame  $w$  can therefore be described by a transformation  ${}^w\mathbf{T}_r$  instead of its pose vector  ${}^w\mathbf{r}$ :

$${}^w\mathbf{r} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \rightarrow {}^w\mathbf{T}_r = \begin{pmatrix} \cos \theta & -\sin \theta & x \\ \sin \theta & \cos \theta & y \\ 0 & 0 & 1 \end{pmatrix}. \quad (3.4)$$

This makes it more easy to describe relative movements from one pose to another, or to transform points from the robot's reference frame into the world frame. In case the robot's current pose is at  ${}^w\mathbf{r}_i$  and it moves to pose  ${}^w\mathbf{r}_j$ , both poses can be described as  ${}^w\mathbf{T}_{r_i}$  and  ${}^w\mathbf{T}_{r_j}$ , respectively. In order to get the relative transformation  ${}^{r_i}\mathbf{T}_{r_j}$ , which describes the pose  $\mathbf{r}_j$  in the reference frame  $r_i$ ,

$${}^{r_i}\mathbf{T}_{r_j} = ({}^w\mathbf{T}_{r_i})^{-1}({}^w\mathbf{T}_{r_j}) \quad (3.5)$$

has to be calculated. The presented notation so far is for the 2D case, where there are three DoF. Analogously, this can be performed for the 3D case as well. The difference is the extension to six DoF, namely  $\mathbf{r} = (x, y, z, \theta_x, \theta_y, \theta_z)^T$ , and a resulting transformation Matrix  $\mathbf{T} \in \mathbb{R}^{4 \times 4}$ . In the present work, the transformations in the six DoF case is only needed for transforming data from the camera coordinate system into the robot system. All cameras are considered calibrated and the respective transformations are already known. Therefore, a further description of the six DoF case is omitted.

### 3.3 Range Sensors

There exist a variety of sensors in order to acquire 3D data. It is, among others, possible to use stereo triangulation, structured light patterns, or also Time

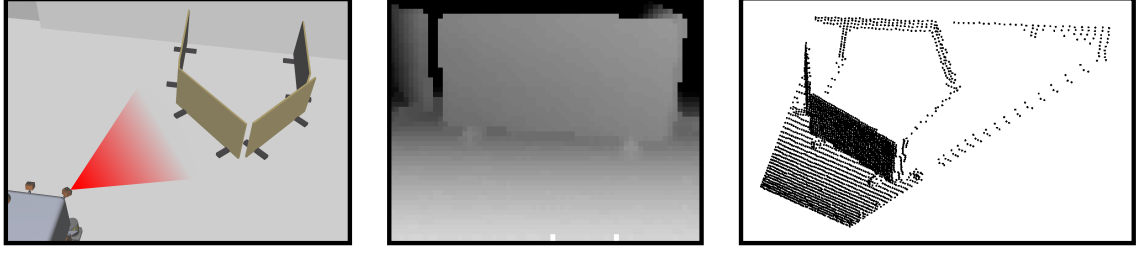
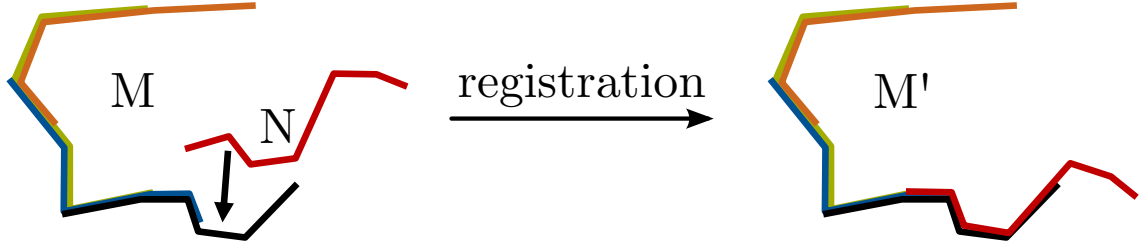


Figure 3.4: Exemplary ToF camera range measurement. In the left picture the recording camera is depicted with the FoV visualized in red. The middle picture represents the depth image from the sensor. On the right side, the resulting point cloud is depicted.

of Flight (ToF) sensors. The Microsoft Kinect is one example, which projects a structured light pattern into the environment and measures the distortion. Another approach are the more expensive laser range sensors called Light Detection and Ranging (LIDAR), where a laser beam is sent into the environment and the *time-of-flight* reflection is measured. Furthermore, it is possible to use stereo images to extract 3D data from regular pictures, in case the relative pose is known.

The used range sensors in this thesis are ToF cameras for the perception of 3D range data. They compute the distances by measuring the travel time of the light, which first is generated by a source, then travels to the observed target, is reflected and then detected by a sensor. The light source and sensor are normally located next to each other in the camera. In Figure 3.3, a range measurement is depicted for the 2D case, where only one line of pixels is used. The individual distance values can be used, in order to calculate a specific point in space. In the 3D case, where a sensor is used with the pixel indexes  $(i, j)$ , a measurement represents a 3D point set  $\mathbf{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_k\}$  and  $k = i \cdot j$ . This point set can be generated based on the known camera geometry. The respective distance value  $d_k$  combined with the pointing direction  $\mathbf{v}_k = (\Delta x_k, \Delta y_k, \Delta z_k)^T$ , where  $\|\mathbf{v}_k\| = 1$ , is used in order to generate the 3D point  $\mathbf{p}_k = d_k \mathbf{v}_k$ . The pointing direction changes for every pixel and is oriented from the pixel location towards the focal point of the camera.

The process of generating such a *point cloud* is exemplified in Figure 3.4. On the left side, a ToF camera is shown in the used simulator, pointing towards an object. The resulting data is depicted in the middle, where a dark pixel indicates a high and bright a pixel a low distance measurement. Combining the distance measurements with the camera geometry parameters makes it possible to generate a point cloud, which is visualized in the right image. Every point in this image is defined by the vector  $\mathbf{p}_k = (x_k, y_k, z_k)^T$ .



### 3.4 Multiview Range Image Registration

In this thesis, SLAM is formulated in terms of *multi-view range image registration*. Thereby, the successive range images are combined together into one common reference frame [CM92]. One single range measurement forms a local reference frame, with the sensor pose as its origin. A multiple of these measurements can not just be combined without further knowledge about their relative position and orientation towards each other. The problem of aligning the distinct pieces of range data into one single reference frame is called *registration*. In the case of two data sets, where  $\mathbf{M}$  is the model and  $\mathbf{N}$  the scene set, the goal is to find a transformation  $\mathbf{T}$ , that, applied to the scene set  $\mathbf{N}$ , aligns the two data sets correctly.

The difficulty is finding an error function, which, if minimized, represents a *best fit* of the data sets. In Figure 3.5, an exemplary registration process is depicted.  $\mathbf{M}$  already consist of multiple combined data sets. The new data set  $\mathbf{N}$  is aligned by registration onto  $\mathbf{M}$  and they form the data set  $\mathbf{M}'$ .  $\mathbf{T}$  is expressed by a rigid transformation, which consists of a translation and rotation. If the robot movement is restricted to the planar case, only three DoF have to be considered. The overlapping of the respective data sets also determines the likelihood of finding a correct alignment. In case they do not overlap at all, it is not possible to calculate any spacial relation between them. In case of overlapping, one difficulty is finding the correct *data-association*. This defines the selection of pairs in the respective data sets, which correspond to the same feature in the real world. A very popular solution for solving this problem is the Iterative Closest Point (ICP) algorithm. It was introduced by Besl and McKay [BM92] and is further described in section 3.6.

There are different approaches for registering several range images into one common reference frame:

1. **Sequential registration:** Here, a set of consecutive range images  $\mathbf{N}_{1:k}$  is registered in a pairwise fashion. The origin of the global reference frame is defined by the reference frame of the first image  $\mathbf{N}_1$ . All further images are registered relative to this reference frame in pairs, which means, image  $\mathbf{N}_k$  is aligned onto  $\mathbf{N}_{k-1}$  for  $k = 2 \dots k$  in order to generate one common model. This method is not very robust because of the accumulation of alignment errors,

and therefore is unlikely to generate a consistent registration [Pul99]. Very small alignment errors can lead to large errors over time and inconsistencies may occur. Especially, in case *loops* are closed, which means that image  $\mathbf{N}_i$  overlaps with image  $\mathbf{N}_j$  and  $j \gg i$ . This kind of pairwise registration is normally followed by global relaxation techniques and graph optimization algorithms [SNLH09].

2. **Metaview registration** In contrast to the sequential registration technique, *metaview* registration uses the complete merged information in order to align any new image. Introduced by Chen and Medioni [CM92], the combination of the aligned data into one reference frame form the so called *metaview*. A new data set  $\mathbf{N}_k$  is registered and integrated into the metaview  $\mathbf{M}_{k-1}$  and forms the new data set  $\mathbf{M}_k$ .

Compared to the sequential registration method, the chance of a correct alignment increases. The registration errors still can accumulate, but because of the integration of all previous data into the registration process, these errors are more likely to be corrected. For smaller environments, like one single room, this method is applicable, but will still result in divergences for large-scale settings. In case a loop is detected after a long traveling distance, the loop losing is unlikely to be correct without further global relaxation steps.

3. **Simultaneous registration** The shortcomings of the two previously presented methods result from the fact, that a measurement  $\mathbf{N}_k$  does not change the previously registered data sets  $\mathbf{N}_{0:k-1}$ . But a new observation of the environment is very likely to change the understanding of some previously gathered data. The idea behind simultaneous registration, is a registration process, which considers all range images at once in order to build a consistent model of the environment. For registering a new data set, all previously gathered data is considered and changed as well. This results in a better and more accurate model, with the drawback of a much higher processing complexity.

In this thesis, the metaview registration method is used. The advantage to the sequential registration is the reduced error accumulation for multiple recordings in the same environment. The simultaneous registration would reduce the alignment error even more, but with the cost of a complete remodeling of the whole existing map representation. This step is computationally expensive, but might be necessary in some cases. But the remodeling by range scan alignment after every exploration step is computationally too expensive for a reasonable and timely exploration procedure.

## 3.5 Map Representation

In order to react to an environment, a robot has to create an internal representation or *map* of it. Somehow, the information from the *real world* has to be made

processable in order to form a basis for further interaction. There are basically two distinctive kinds of environment representations:

1. **Egocentric Map:** In this kind of environment representation, the robot forms the origin. All gathered data and features of the environment are stored in relation to the robot's position and orientation. In case the robot moves from one place to another, the position and orientation of all gathered features has to be transformed as well, in order to retain an *egocentric* map.
2. **Allocentric Map:** This forms the counterpart of the previous representation, where all data and features are expressed in relation to a *global reference frame*. In case the robot performs a movement, all features remain on their position and only the robot changes its position and orientation in the reference frame.

In addition, a map is used to represent a geometric structure, which is normally expressed by a *metric map* representation. Its counterpart, the *topological map*, in which only qualitative relations and descriptions are stated between places, has fallen out of fashion in the recent years [SK08, p. 874]. More often, hybrid approaches combining both are used either to connect allocentric maps or divide them into parts in order to form nodes [TBF05]. The most common realizations of *metric maps* are:

1. **Continuous Maps:** They are also called *feature maps*. In this kind of map, all data is stored by a continuous range of values. These features can describe various things, like points in space, individual distance measurements, or also whole objects like walls or chairs. The main disadvantage for this representation is the growing amount of necessary memory data over time. In case no involved reduction and segmentation algorithms are performed, the amount of data can easily become infeasible for further processing. In addition, it is often difficult to determine a separation between free and occupied space.
2. **Grid Maps:** Also called *discrete maps*. They use a discrete range of values in order to describe an environment. Normally realized as *occupancy grid maps*, introduced by [ME85], one grid cell represents the state of a discrete spacial unit. It can be encoded binary into *free* or *occupied*, or into a more probabilistic representation, in which the state of a cell describes the probability of being either *free* or *occupied*. The difficulty for grid maps is to find a reasonable trade-off between grid resolution and memory efficiency. The larger the grid size, the easier it is to process and store the data, but the fewer details can be represented.

In the following, the problem of construction an environment representation, with already known poses of the robot is regarded. The localization of the robot in a global reference frame is considered to be known, and the process of generating an appropriate map out of the collected sensor data is addressed. In this thesis, two different representations are used, namely, a *point map* and an *occupancy grid map*.

### 3.5.1 Point Maps

Point maps are a simple and straight forward way of representing an environment. They are strongly related to raw sensor data from range images and the measurements already form a representation of 2D or 3D point sets in space. From a sequence of measurement, the corresponding point map can directly be calculated, without further feature extraction or other additional processing steps. One observation of the environment is called *point cloud* and the data is perceived relative to the local reference frame  $s$  of the sensor. The important processing step is to describe these point sets relative to the robot pose at the respective recording time.

For clarification, the used notation is based on [SK08], where  ${}^s\mathbf{p}$  defines a point vector in the reference frame  $s$ .  ${}^r\mathbf{T}_s$  defines an affine homogeneous transformation between the reference frames  $s$  and  $r$  by Equation 3.6.

$${}^r\mathbf{p} = {}^r\mathbf{T}_s {}^s\mathbf{p} = \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{homogeneous transformation}} {}^s\mathbf{p} \quad (3.6)$$

${}^r\mathbf{T}_s$  represents a  $4 \times 4$  homogeneous rigid body transformation and allows the combination of rotation and orientation into one matrix multiplication. The transformation matrix  ${}^r\mathbf{T}_s$  from the sensor reference frame  $s$  to the robot frame  $r$  is defined by the extrinsic calibration parameters. A previous calibration process, which is not in the scope of this thesis, determines this transformation. Only if the relative sensor position or orientation is changed, the transformation matrix values change. For several sensors, each sensor pose is described by its own transformation  ${}^r\mathbf{T}_{s_i}$ .

The previous transformation makes it possible to represent all gathered range data relative to the robot reference frame. With the help of the known robot poses, the different data sets can be combined into one global reference frame, called the world frame  $w$ . Therefore, another rigid body transformation  ${}^w\mathbf{T}_{r_i}$  has to be applied to the point cloud, which differs at every distinct robot pose.

$${}^w\mathbf{p} = {}^w\mathbf{T}_{r_i} {}^r\mathbf{p} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 & t_x \\ \sin \theta & \cos \theta & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} {}^r\mathbf{p} = {}^w\mathbf{T}_{r_i} {}^r\mathbf{T}_{s_j} {}^s\mathbf{p} \quad (3.7)$$

In Equation 3.7, the transformation  ${}^w\mathbf{T}_{r_i}$  is already represented for a robot movement on a plane. The 3D case with 6 dDoF is not considered in this thesis, because the robot only moves within a planar environment and the extrinsic camera parameters are known. After the application of the correct transformations, all point clouds can be modeled in the common world reference frame. One big disadvan-

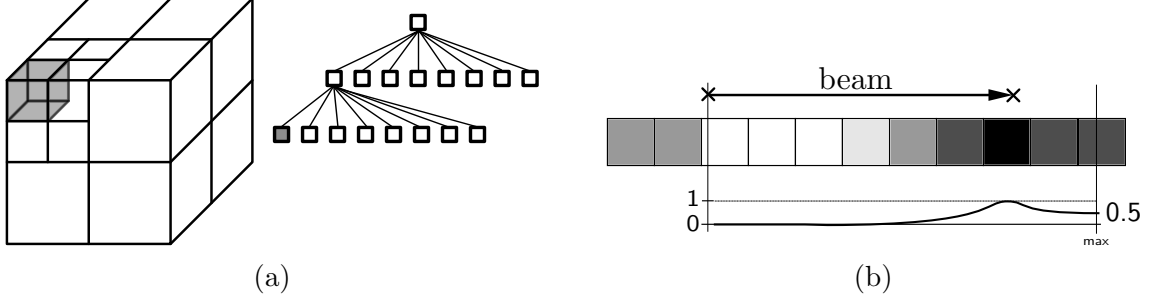


Figure 3.6: Octree data structure (a) and an update step for the occupancy grid map (b). The darkness of a cell indicates its occupancy probability.

tage of this modeling technique is the missing distinction between *free* and *occupied* space. Hence, the following representation is used as well.

### 3.5.2 Occupancy Grid Maps

With grid maps, the real world environment is divided into equally sized units of space called *grid cells*. Such a grid cell represents a description for the knowledge about this particular space. As a grid map representation, *occupancy grid maps*, introduced by Moravec and Elfes [ME85] are often used in exploration approaches. The idea behind them, is to describe each cell with its own value, expressing the probability of being occupied by any obstacle. Each of such a grid cell possesses a probability of being free or occupied. Cells which have not been observed yet usually get the initial value 0.5. They are also referred to as being *unknown*. The big advantage of such a grid map is the distinction between *occupied* and *free* space, which is not possible by a point map. Additionally, they allow a constant time access. The discretization errors and high memory requirement, depending on their resolution, are drawbacks which have to be taken into account.

These grid maps are also applicable to 3D representations of the environment. To decrease the impact of the exponential growth of data, smart data structures like **octrees** are used to represent the map. It is a tree-based hierarchical representation, in which each grid cell is subdivided into eight smaller cells. The leafs, as a smallest division unit, are also called voxel (from volume/pixel). In case all eight sub-units of a node contain the same values, it is only necessary to store this information in the node itself with one single value. This enables a memory efficient way of representing 3D grid maps, where much space has the same state (either *unknown* or *free*). In Figure 3.6a, such an octree is shown exemplarily. Representing the grid map by  $\mathbf{m} = \{m_1, m_2, \dots, m_k\}$ , the robot poses by  $\mathbf{X}_{1:t} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_t\}$  and measurements by  $\mathbf{Z}_{1:t} = \{z_1, z_2, \dots, z_t\}$  the occupancy grid mapping approach represents the posterior probability  $p(\mathbf{m} | \mathbf{x}_{1:t}, \mathbf{z}_{1:t})$ . Grids cells  $m_i$  are normally also considered independent



to keep the calculations tractable:

$$p(\mathbf{m} | \mathbf{X}_{1:t}, \mathbf{Z}_{1:t}) = \prod_{i=1}^k (m_i | \mathbf{X}_{1:t}, \mathbf{Z}_{1:t}) \quad (3.8)$$

Updating the occupancy probability of a cell strongly depends on the used sensor. An inverse sensor model is needed, which specifies the probability of a cell of being occupied after a sensor reading. For further information, [Sup08] can be referred to, where different update strategies are described in more detail. Such a 3D grid space, or also called *voxel space*, can be updated after various fashions, whereby in this thesis the Bayes-Update is used. It is performed in a ray-tracing fashion, where beams are generated from a sensor measurement and integrated into the 3D voxel space.

The individual steps of integrating a sensor measurement into an occupancy grid map are described in the following:

1. The measurement is represented relative to the world reference frame  $w$ . This is performed equally to generation of the point cloud map.
2. Starting from the origin of the sensor frame, so called *beams* or *rays* are created, which point from the origin to the particular sensor readings including the respective distance.
3. The beams are integrated into the occupancy with the chosen update strategy. The parameters of the sensor reading are also very important for this step, because they define the certainty of a measurement of being correct. In Figure 3.6b, an exemplary update step for one dimension is depicted, where a beam is integrated into the occupancy grid map. The gray values indicate the probability of a cell of being occupied. The dispersion of the end point resembles the variance parameter of the real sensor in order to model the accuracy as good as possible. This step is performed in the 3D occupancy grid map in a ray-tracing manner, where all intersecting voxel are processed.

### 3.5.3 Collision Space in Grid Maps

Based on the information given in [GBL02] a *safe region* is necessary for the exploration process. Somehow, a distinction has to be made between the region, where the robot can drive around without colliding with any object, and the region, where it is not allowed to go. This forbidden areas might contain obstacles, like walls, chairs or tables, or also obstacles like the descent of a stairway. By the help of the sensor measurements, it is not directly possible to judge, where the robot is able to go and where it is not. But there is knowledge about the positioning of the sensors and also their orientation in relation to the robot. A method has to be found, in order to process the gathered data, and distinguish between the traversable and

forbidden space. In the present work, a collision radius is applied to the occupied cells, which keep the robot from colliding with them. The exact process is described in subsection 4.5.2.

## 3.6 ICP Range Image Registration

In order to create a consistent model out of the sensor measurements, the distinct data sets have to be merged into one common reference frame. This process is called registration. In case the robot poses from the odometry reading were already exact, the point sets could be combined, like described in subsection 3.5.1. But they are prone to errors and need correction. The ICP algorithm is one popular way for solving this problem and will be briefly described in the following.

### 3.6.1 The Algorithm

Given two point sets  $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_k\}$  and  $\mathbf{N} = \{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_l\}$ , we want to find the rigid body transformation consisting of the rotation  $\mathbf{R}$  and translation vector  $\mathbf{t}$ , which minimizes the following cost function (from [Nüc09]):

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^k \sum_{j=1}^l w_{i,j} \|\mathbf{m}_i - (\mathbf{R}\mathbf{n}_j + \mathbf{t})\|^2. \quad (3.9)$$

If the  $i$ -th point of  $\mathbf{M}$  and the  $j$ -th point of  $\mathbf{N}$  correspond to the same point in space, the value for  $w_{i,j}$  is 1. Otherwise it is 0. The first challenge is to find the corresponding points. To find the optimal transformation  $(\mathbf{R}, \mathbf{t})$ , which minimizes the error function, is the next important step. In [BM92], it is proven, that the iterative algorithm terminates in a local minimum, provided, the corresponding points in the last step are assigned correctly. It is represented in Algorithm 1.

Because of the iterative nature of the algorithm, the guess of the initial transformation is very important for the convergence process. If the guess is too far away from the optimal solution, or if there is too little overlapping, the whole process will result in a failure, or it may converge to a wrong local minimum. Therefore, the odometry readings from the robot serve as an initial estimate in order to combine the distinct data sets.

Furthermore, the robot is restricted to movements on a plane, and therefore the rigid body transformations are also restricted to three Degrees of Freedom (DoF). The basic ICP algorithm calculates rigid body transformations for the 6 DoF case, therefore, the algorithm of finding the transformation  $\mathbf{R}$  and  $\mathbf{t}$  can be restricted. This reduces the search space and enables a faster calculation of the solution.

---

**Algorithm 1** The ICP algorithm [Nüc09]

---

```
1: for  $i = 0$  to  $maxiteration$  do
2:   for all  $\mathbf{n}_j \in \mathbf{N}$  do
3:     Find the corresponding closest point within a range  $d_{max}$  in the set  $\mathbf{M}$ 
       for every point  $\mathbf{n}_j$ .
4:   end for
5:   Calculate the transformation  $(\mathbf{R}, \mathbf{t})$ , which minimizes the error function from
       Equation 3.9.
6:   Apply the transformation from step 5 to data set  $\mathbf{N}$ .
7:   Compute the difference in the quadratic error  $E_{i-1}(\mathbf{R}, \mathbf{t}) - E_i(\mathbf{R}, \mathbf{t})$  before
       and after the application of the transformation. Terminate, if the difference
       is smaller than a certain threshold  $\varepsilon$ .
8: end for
```

---

### 3.6.2 Overlapping and Noise

The algorithm assumes, that for every point  $\mathbf{m}_i$  in  $\mathbf{M}$  exists a corresponding point  $\mathbf{n}_j$  in  $\mathbf{N}$ . If this is not the case, false pairs can negatively affect the outcome. A limiting search radius can restrict the error. There are two types of incorrect correspondences: *initially false* and *generally false* ones. A pair is initially false if  $\mathbf{m}_i$  and  $\mathbf{n}_j$  are selected as pairs, but  $\mathbf{m}_i$  actually corresponds to another point  $\mathbf{n}_k$  with  $k \neq j$  because of a poor initial estimate. After several ICP iterations  $\mathbf{m}_i$  is finally paired correctly with  $\mathbf{n}_k$  and the delivered solution is correct. However, if too many pairs are initially false, the algorithm may directly converge to a false local minimum. If a pair is generally false there is no corresponding point at all. This can be caused by noise or a non complete overlapping of the respective data sets. Noise can not be avoided, and the data sets can not always overlap completely, otherwise no exploration of new regions is possible.

In order to cope with these problems, a reasonable rejection of specific correspondence pairs has to be found, e.g. a limited range between the corresponding points [Zha00], or using only a certain subset of the data set  $\mathbf{N}$ . In this thesis a limited search radius is applied for the ICP algorithm. This reduces the effect of missing correspondences in regions, where distinct data sets do not overlap.

## 3.7 Entropy-based Exploration

The exploration strategy in this thesis is based on a combination of frontier- and entropy-based exploration. The frontier serves as an initial analytic strategy, in order to minimize the region of possible target locations. In addition, a zig-zag movement between distant borders is suppressed. But the main goal is to reduce the uncertainty of all observable cells in classifying them into either being free or

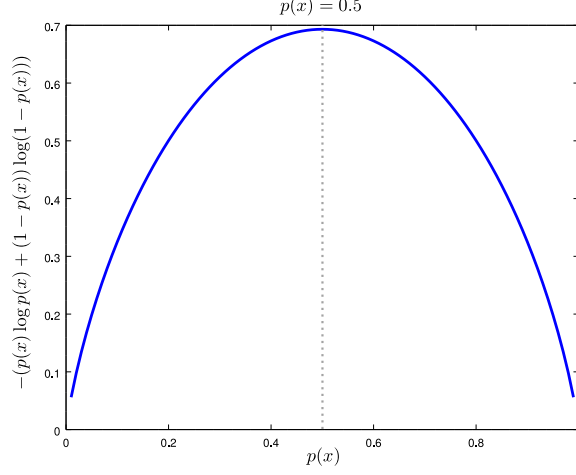


Figure 3.7: The occupancy grid entropy for a single cell.

occupied. In other words, this task states the maximization of information gain, or the reduction of entropy during the exploration [VAC13].

A common measurement for the uncertainty is the entropy  $H(x)$  [TBF05, ch 17].

$$H(x) = E[-\log p(x)] = \int_x p(x) \log p(x) \quad (3.10)$$

It is the expectation value of the information content of the variable  $x$ . The entropy of a cell  $m_i$  in an occupancy grid map with the occupancy probability  $p(m_i)$  is computed by Equation 3.11.

$$H(m_i) = -(p(m_i) \log p(m_i) + (1 - p(m_i)) \log(1 - p(m_i))) \quad (3.11)$$

In Figure 3.7, this function is visualized. For the occupancy probability values of 0 or 1, the entropy is minimal, whereas it is maximal for highest uncertainty with  $p(m_i) = 0.5$ . The entropy basically is a measurement for the present information about the environment. The more the entropy can be reduced, the higher is the certainty about the state of the occupancy map. The idea of the entropy-based exploration is to express the information gain  $I(t, t + 1)$ , after executing an action from time step  $t$  to  $t + 1$  with Equation 3.12.

$$I(t, t + 1) = H(x_t) - H(x_{t+1}) \quad (3.12)$$

The sensory information at the robot pose  $x_{t+1}$  is integrated into the map in order to calculate  $H(x_{t+1})$ . To apply the entropy criteria to the exploration task, a virtual measurement is performed to estimate the information gain for a specific action. The procedure for a virtual measurement is briefly described in the following.

## Simulated Measurement

A range scan is simulated by a ray-casting procedure, where a beam originating from the camera center is only intersected by occupied space. For free or unknown space, it will not be intersected and continues its path through the occupancy grid map until the beam reaches a maximum length. Therefore, this simulation procedure gives an upper estimate about the possible information gain of a robot pose configuration.

All cells, which are influenced by the simulated range measurement are considered for calculating the entropy  $H(x_{t+1})$ . The individual entropy values of each cell  $m_i$  are simply summed up by  $H(x_{t+1}) = \sum_i H(x_i)$ . The more unknown space is likely to be perceived by a robot pose  $x_{t+1}$ , the higher the estimated information gain will be. Of course, this estimation will not always be correct, because the outcome of the real measurement is not yet known. But it gives a good estimate about which robot pose might lead to the highest information gain.

Like mentioned in section 2.4, the analytical determination of the next robot pose with the highest entropy reduction is related to the art gallery problem and infeasible for a practical use. Therefore, a randomized strategy is used, where individual samples are generated and evaluated.

# Chapter 4

## Autonomous Exploration Process

In this chapter, the whole autonomous exploration process will be discussed, focusing on its realization in software. Every algorithm is written in C++ and integrated into the institute’s internal L3D-library. In the following, the implementation environment and project specific details will be explained. Afterwards, the process steps which need to be executed in order to fulfill the autonomous exploration task are described.

### 4.1 Mobile Robot Environment

The relevant software and hardware tools for this thesis are further described in this section. The target platform is the KUKA omniRob, which is additionally equipped with eight ToF cameras. The software development is realized solely within the MRE simulation environment and by the use of L3D internal tools.

#### 4.1.1 OmniRob Platform

The omniRob (Figure 4.1a) is a robot supplied by KUKA. Two laser range sensors for 2D measurements are initially installed on the robot, but they are not used for the implemented algorithms in the present thesis. For the 3D mapping purpose, eight ToF cameras are additionally attached on top of the robot and make it possible to perceive a large field of view. The developed algorithm in this thesis is self-contained and able to run on other mobile robot platforms as well. It only depends on data from 3D range sensors and the robot’s odometry data.

The specifications for the omniRob platform are still important, because the algorithm is targeted for the deployment on the robot. Furthermore, the MRE simulation environment uses the robot’s parameters.



(a) The KUKA *omniRob* platform.



(b) ToF camera O3D300.

Figure 4.1: The used omniRob platform at the DLR-Institute in Oberpfaffenhofen (4.1a). Only the eight ToF cameras, attached around the robot, are used for the integrated exploration. One single ToF camera is depicted at 4.1b (source: DLR-intern)

Some important specifications are:

- size (length  $\times$  width  $\times$  height): 1 200 mm  $\times$  712 mm  $\times$  645 mm
- weight: 270 kg
- maximum speed: 1 m/s

Furthermore, the robot is able to move omnidirectional with the help of its mecanum wheels. The robot's computation unit consists of four I7 processor boards.

#### 4.1.2 O3D100 Photonic Mixing Device

In this thesis, eight O3D100 Photonic Mixing Device (PMD) 3D sensor cameras are used. In contrast to a laser range sensor, these ToF cameras do not use a single laser, but illuminate the whole scene at once by infrared light. They have an infrared Light-Emitting Diode (LED) integrated with a wavelength of 850 nm, which is not in the visible spectrum. The sensor is able to measure the phase shift for every pixel at the same time.

The panoramic arrangement of the eight ToF cameras allow an almost complete surround view of the environment, but there is still space around the robot which can not be perceived. In Figure 4.2b, the *FoV* of one single camera is depicted exemplarily from a side perspective. It is not possible for the robot to register the obstacle in red, because it is located in its blind angle. Additionally, space above

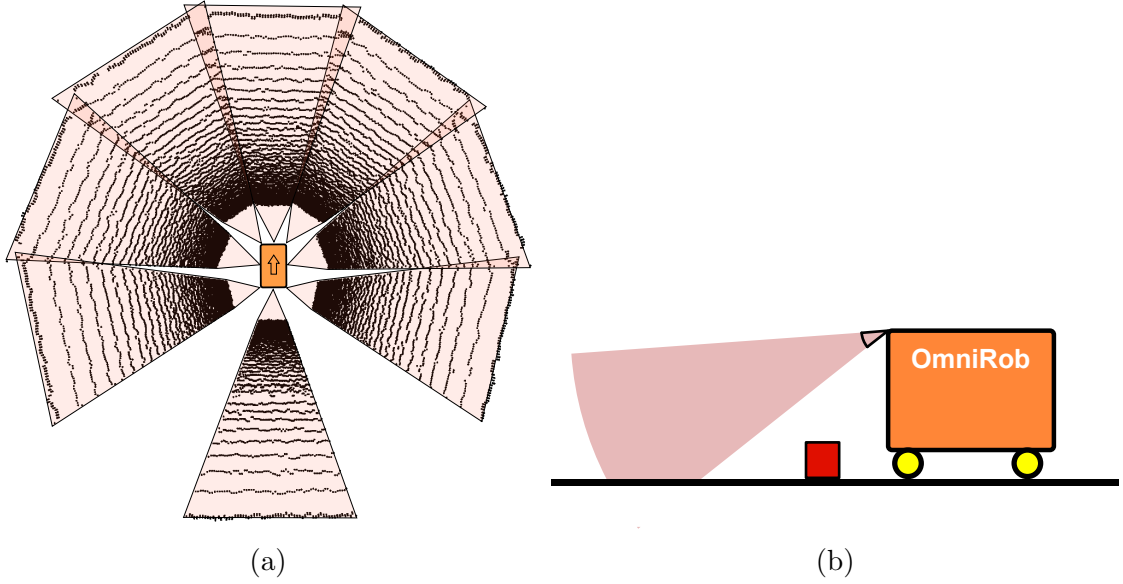


Figure 4.2: OmniRob FoV visualization. A measurement is depicted in (a) with the robot standing on a wide plane without obstacles. The black dots indicate the calculated depth image points on the floor. In (b), the FoV of one single camera is depicted sideways.

the height of 60 cm can not be perceived either with the current camera setting. But this is still sufficient to create an environment representation which can be used for further processing steps. The O3D100 PMD has the following parameters:

- The opening angel is  $40^\circ \times 30^\circ$  .
- The resolution in pixel is  $64 \times 48$
- The measurement frequency is 20 Hz
- The measurement range is from 12 cm to 7 m

One perception of the environment results in  $64 \times 48 = 3072$  points per camera. Therefore, 24576 points are generated by one measurement step with all eight cameras combined.

In this thesis the calibration process is omitted and all ToF cameras are expected to be positioned and calibrated correctly. For more information about the calibration process of ToF cameras, the work from Fuchs S. et al. [FM08] is referred to. It implies that the pose of every camera  $\mathbf{r}_{cam,i}$  relative to the robot pose  $\mathbf{r}_{robot}$  in the robot reference frame  $r$  is known.

Every camera has its own reference frame  $c_i$ . Within the calibration process, the extrinsic parameters of the camera are determined. The targeted output is a transformation matrix  ${}^rT_{c_i}$  for each camera. The eight transformation matrices are expected to be known in the present work. This enables the calculation of one common



point cloud in the robot reference frame. The process of generating such a *point cloud* is described in section 3.3 and exemplified in Figure 3.4.

### 4.1.3 L3D C++ Library

The C++ Library Lib3D (L3D) is used in order to solve all robot tasks, mentioned in this thesis. The library is developed at the German Aerospace Center (*"Deutsches Zentrum für Luft- und Raumfahrt"*) (DLR) Institute of Robotics and Mechatronics and is still under development. It provides many useful tools for the solution of robot and 3D perception relevant problems. The most important tools used in this thesis where the ICP algorithm and the Dynamic Octree Voxel Space, which includes useful update procedures for range data sets. Furthermore, noise models for the odometry and range measurements, based on the manufacturer data, are included. This L3D probabilistic octree space is developed and maintained at the DLR institute. It is very similar to the OctoMap [WHB<sup>+</sup>10], which is released under the *BSD 3-clause* license. The used ICP algorithm is able to work with constrained rigid body transformations, where only a translation on a plan and a rotation around the *z*-axis are considered.

### 4.1.4 Simulation Environment

For the purpose of fast and efficient software development for the KUKA omniRob, an institute intern simulation environment called Mobile Robot Environment (MRE) is available. It is also possible to visualize the robot and create test maps for it with 3D computer graphics software like Blender. In Figure 4.3, the FoVs of all cameras are visualized for the simulated robot.

With the simulator, it is possible to model odometry and sensor noise, similar to reality. Furthermore, the MRE software is used with the same interfaces as the real robot, so there is nearly no difference between the real robot and the simulation of it. This simulator was written to enable the developers a fast testing of their software, without the need of starting the whole robot system and setting up a new environment.

The simulation tool consists of different parts, where the following three parts are the most important ones for this thesis:

1. Simulation of the ToF and laser sensors. The simulation of these sensors use a polygon-based model of the environment and calculates the distance from the position of the sensor to the next polygon. A gaussian model is used to add noise to this distance. The gaussian model is based on the noise model given by the manufacturer.

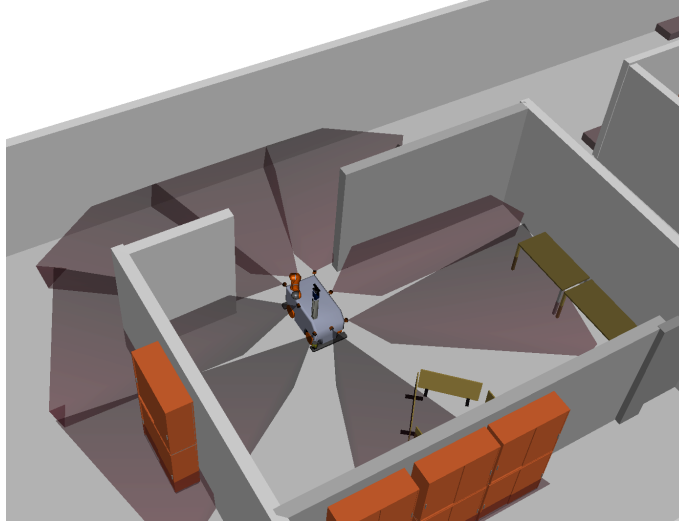


Figure 4.3: The omniRob in the MRE simulator with the FoV of all cameras visualized. (The visualization does not resemble the actual measurement, because the camera can not measure through walls.)

2. Simulation of the mobile platform. This tool simulates the movement of the robot. The simulated motion depends on measurements, which were made to determine the precision of the robot's translation and rotation. These motion noise models are used to simulate the robot as realistically as possible.
3. Visualization of the omniRob. The viewer enables a quick overview over the actual status of the system. In Figure 4.3, the simulated robot is depicted exemplarily.

## 4.2 System Overview

The focus of this thesis is on a combination and integration of a SLAM and exploration process. In Figure 4.4, the whole autonomous exploration process workflow is depicted. The sensors on the left side are for one thing the eight ToF cameras and for another thing the odometry sensor. They form the connection to the physical world in being able to perceive it. Their data serves as an input to the SLAM process. During the execution of this process, a model of the environment is generated and represented as a 3D occupancy grid map (subsection 3.5.2), by *metaview* registration (section 3.4). The used occupancy grid map is realized as an octree data structure, in which each voxel state value represents the probability of being *occupied*.

Based on the occupancy grid map and the robot's current position, the exploration process proposes a next best sensing pose, in order to efficiently and completely

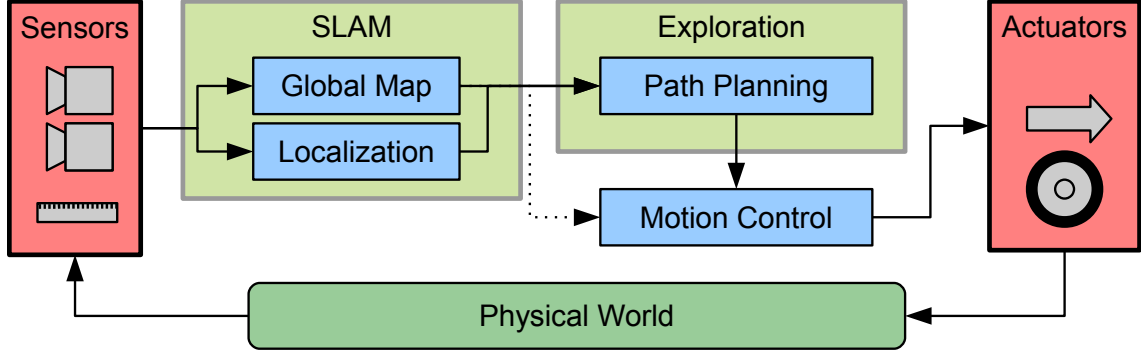


Figure 4.4: Autonomous exploration process workflow.

cover the whole environment. Thereby, the expected information gain with regard to the path distance is evaluated. Once a path has been created, the motion control process ensures, that the robot reaches the target pose. The generated commands from the motion control are sent to robot controller, where they are executed by moving the wheels in the correct manner. A new robot pose in the physical world is the result of this action. During the path execution, the exploration process is not invoked until the target point is reached. Only the SLAM process is needed in order to localize the robot with respect to the internally constructed map. This prevents the robot from colliding with the environment during the path execution. The whole process loop continues until a complete map of the environment is created.

### 4.3 Map Notation

For the subsequent description of the individual exploration process steps, the important map representations are noted in the following way. Their generation and application is further described and illustrated in the regarding section.

**$P$** : The raw point cloud from the range measurement

**$Q$** : The point cloud  **$P$**  with removed floor.

**$M$** : The metascan, as an accumulation of point clouds from  **$Q$** .

**$V$** : The 3D occupancy grid map, as a result from the SlamICP process.

**$G$** : The 2D grid map, as a projection of  **$V$**  onto a plane.

### 4.4 SlamICP Process

When the robot takes a recording of the environment, the resulting depth images and the respective odometry estimate are used as an input for the SlamICP process.

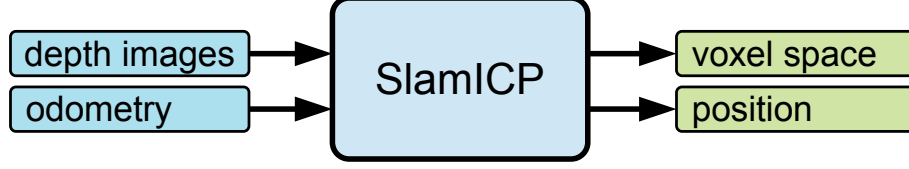


Figure 4.5: The encapsulated SlamICP process. The depth images and the current odometry reading serve as an input. A 3D occupancy grid map and the robot pose in this map are the targeted output parameters.

Based on this information, a voxel space is created with the robot's estimated pose in it, as can be seen in Figure 4.5.

The three important steps performed during the SlamICP process are:

1. Robot Pose Estimation
2. Pose Refinement
3. Occupancy Grid Mapping

They are further described in the following.

#### 4.4.1 Robot Pose Estimation

The first estimated robot pose estimate will always be  $\mathbf{r}_0 = (0, 0, 0)$ , no matter what initial odometry reading is perceived. This means, that the first robot pose in the world reference frame  $w$  is described by the transformation  ${}^w\mathbf{T}_{r_0} = \mathbb{I}$ . The first odometry input  $\mathbf{b}_0 = (b_{x1}, b_{y1}, b_{\theta1})$  is not necessarily  $\mathbf{b}_0 = (0, 0, 0)$ . This may result from a previous robot movement, before the exploration algorithm is started. The transformation  ${}^b\mathbf{T}_{b_0}$  describes this pose change. The robot has its own reference frame  $b$  for the odometry readings. For a clarification of the used notation, it is conferred to section 3.2.

In case a new odometry input  $\mathbf{b}_i$  is received, the movement relative to the last input  $\mathbf{b}_{i-1}$  in the  $b_{i-1}$  reference frame is calculated in form of an affine Transformation  ${}^{b_{i-1}}\mathbf{T}_{b_i}$  by:

$${}^{b_{i-1}}\mathbf{T}_{b_i} = ({}^b\mathbf{T}_{b_{i-1}})^{-1}({}^b\mathbf{T}_{b_i}). \quad (4.1)$$

This transformation is used for the estimation of the robot pose  ${}^w\mathbf{T}_{r_i}$  in the internal world reference frame  $w$ :

$${}^w\mathbf{T}_{r_i} = ({}^w\mathbf{T}_{r_{i-1}})({}^{b_{i-1}}\mathbf{T}_{b_i}). \quad (4.2)$$

This calculation is possible, because the relative movements are equal, even if the respective reference frames do not coincide.

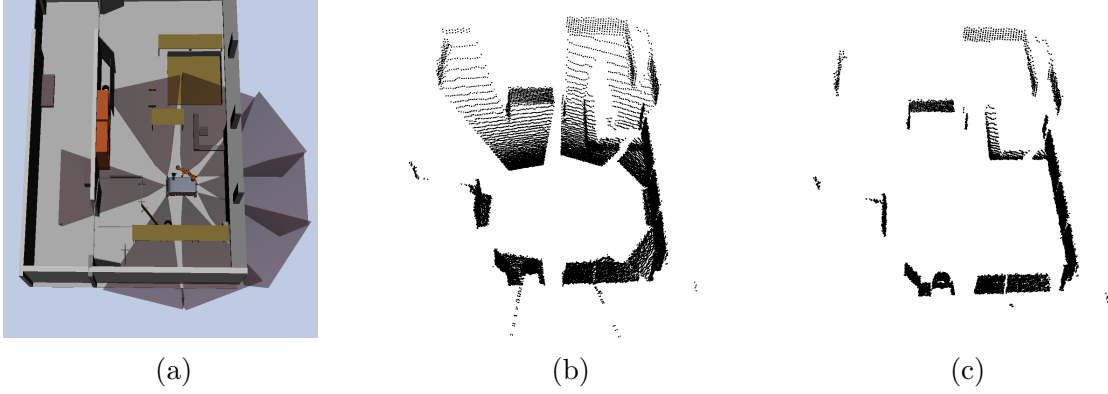


Figure 4.6: Point cloud generation. Figure 4.6b is the raw point cloud from the range measurement performed in the simulator Figure 4.6a. In Figure 4.6c, the floor points have been removed.

#### 4.4.2 Pose Refinement

Based on the estimated robot pose from the last step, the acquired camera recordings are transformed into a point cloud  $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$ . This point cloud resembles the collision points of the individual camera beams, relative to the internal reference frame.

After the generation of the point cloud, the floor points are removed by a specified threshold  $z_{min}$  for the  $z$ -axis. The floor points are not necessary for the ICP algorithm, because only translations and rotations on a plane are considered and furthermore, the amount of data which has to be processed is reduced. There is no vertical shift in  $z$ -axis direction between distinct range measurements, because the robot is only driving on a plane. Typically, the ICP algorithm is applicable for the 3D case with six DoF. Within the L3D library, it is possible to modify the ICP algorithm for the 3D case with only three DoFs. This reduces the possible alignment error and the processing complexity.

An exemplary point cloud generation process is depicted in Figure 4.6. The resulting point cloud without floor points is called  $\mathbf{Q}$ , where  $\mathbf{Q} \subseteq \mathbf{P}$ . This process is executed for all new measurements. In case the robot moves to another pose, the received odometry estimate will not exactly resemble the real world position and orientation. Small errors in rotation and translation will occur. This step is visualized in Figure 4.7a, where the robot turns left and takes a new measurement, but the merged two data sets do not align correctly because of a rotation error.

The ICP scan registration is then used to reduce the odometry-based pose error. The point cloud  $\mathbf{Q}_0$  from time step  $t = 0$  will be directly used to build the metascan  $\mathbf{M}_0$ , because it can not be aligned with any preceding data. In case it is one of the consecutive measurements, the point cloud  $\mathbf{Q}_i$  is aligned by ICP with the metascan  $\mathbf{M}_{i-1}$  and then integrated into it. This results in the new metascan  $\mathbf{M}_i$ .

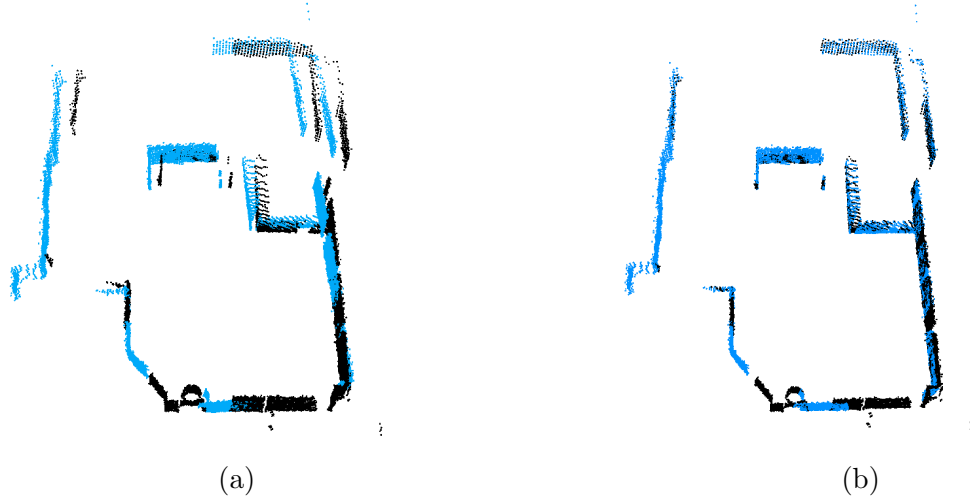


Figure 4.7: ICP range scan alignment of two consecutive data sets. Figure 4.7a is the initial pose estimate, based on odometry data, whereas Figure 4.7b represents the corrected pose estimate after the alignment. The current measurement in blue is matched onto the existing point cloud.

A rigid transformation  ${}^{aligned}\mathbf{T}_{initial}$  is calculated by the ICP algorithm. The transformation is applied to the point cloud  $\mathbf{Q}$  with  $\mathbf{Q}_i' = {}^{aligned}\mathbf{T}_{initial}\mathbf{Q}_i$  and then subsequently integrated into the metascan  $\mathbf{M}$ .

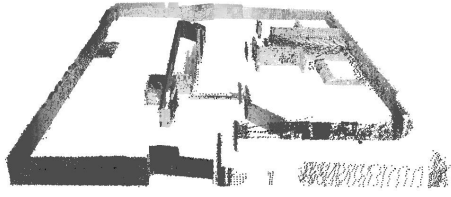
This resulting transformation applied to the estimated robot pose  $\mathbf{r}_i$  from the last step gives a new *refined* pose estimate about the robot.

### Point Cloud Reduction

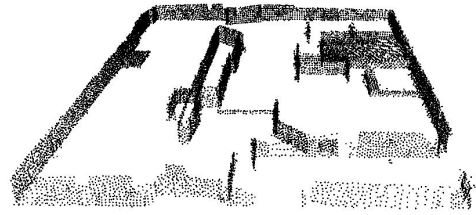
The integration of all subsequent range measurements  $\mathbf{Q}_i$  into  $\mathbf{M}$  will lead to a very dense and large metascan  $\mathbf{M}$ . For the ICP algorithm, a nearest neighbor search is necessary to determine the correspondences. In order to keep this process performant and also reduce the memory requirement, the density of the metascan is reduced.

For every point  $\mathbf{q}_i$  which is integrated into the metascan  $\mathbf{M}$ , a nearest neighbor search is performed. An octree data structure is used for the density reduction, because it enables a fast access to the near surrounding of a specific point. If there already exists a point  $\mathbf{m}_k$ , which is closer to the point  $\mathbf{p}_i$  than an initially specified threshold  $d_{min}$ , the point will not be integrated into the metascan. It is only integrated if  $\|\mathbf{m}_k - \mathbf{p}_i\| > d_{min}$  holds true for all points in  $\mathbf{M}$ . The octree cells, which intersect with a sphere of  $r = d_{min}$  around the point  $\mathbf{p}_i$  are iterated through for the distance search.

The advantage of this procedure is the high memory saving ability. If many consec-



(a)  $M$  with 253210 points with no point cloud reduction. All range scans are simply combined.



(b)  $M$  with 11424 points where point cloud reduction with  $d_{min} = 3$  cm is used.

Figure 4.8: Metascan for a test environment.

utive range scans are performed at the same robot pose, the metascan does hardly increase in size. If no point cloud reduction was performed, many points would be present at the same location in space and this would result in a high increase of computing power for the ICP algorithm. In Figure 4.8, this point cloud reduction is visualized.

### 4.4.3 Occupancy Grid Mapping

After the newly gathered data has been successfully registered into the existing metascan  $M$ , the pose of the robot is known in relation to the internal global map representation. The sensor data can now be integrated into a 3D occupancy grid map, which is called  $V$ . The occupancy grid map is a probabilistic octree space, already available in the L3D library. It is a voxel space, in which the individual entries describe the state of a certain volume. Further information is given in subsection 3.5.2. The integration of sensor data into the map is very similar to the process of *ray casting*, where the first intersection of a ray with an object is determined.

The most important parameter for the map  $V$  is its grid resolution  $l_{res}$ . One voxel describes the state of a volume with the size of  $l_{res} \times l_{res} \times l_{res}$  in the real environment. In this thesis, resolutions from 2 cm to 10 cm edge length are tested. A smaller resolution allows the more precise representation of map details, with the drawback of higher memory and processing power consumption.

The space around the robot's starting pose is initially defined as free, in order to avoid the presence of unknown space on the floor around the robot. Based on the robot's geometry and the positioning of the ToF cameras, a space with the edge lengths of  $2,5\text{ m} \times 2,5\text{ m} \times 0,5\text{ m}$  is initially defined as free. This space is centered around the robot and is standing on the ground plane. Without the preinitialized



Figure 4.9: 3D occupancy grid map with a resolution of 5cm edge length. The occupancy value of a voxel is described by its color, where black indicates a high probability of being occupied. Free space is invisible.

free space, the robot is not able to move to any other place, because it is impossible for it to perceive the vicinity on the floor near the robot. In Figure 4.2b, this fact is visualized.

The integration of the range scan is performed for each camera individually. First, the position and orientation of the camera in the 3D space is calculated relative to the world reference frame  $w$ . Then, for each pixel value, a *beam* is defined, with the starting pose at the camera center. The direction points from the pixel location towards the focal point of the camera. This process is described more exactly in 3.5.2.

The beams are integrated into the 3D occupancy grid map by a *bayes update* rule. The exact update process is described in [Sup08]. Internally, this is performed by the addition of a likelihood factor in log-odds representation. The sensor noise parameters and previous cell states are considered for the resulting occupancy values of the grid cells. In Figure 4.9, an exemplary occupancy grid map is depicted. The space cells below a certain threshold are defined as free and invisible in the picture. The darker a voxel is depicted, the higher its probability of being occupied. They resemble walls, tables or other obstacles, which restrict the possible movements of the robot.

## 4.5 Exploration Process

The exploration process also has narrowly defined interfaces, as can be seen in Figure 4.10. The 3D voxel space  $\mathbf{V}$  generated by the SlamICP process and the robot's estimated pose in it are used as inputs. Based on this information, a path



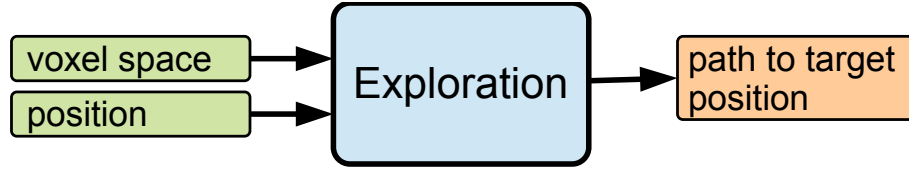


Figure 4.10: The interfaces of the exploration process. Only the voxel space and the robot’s current pose are needed as input. The result of this process will be a path to a new target pose.

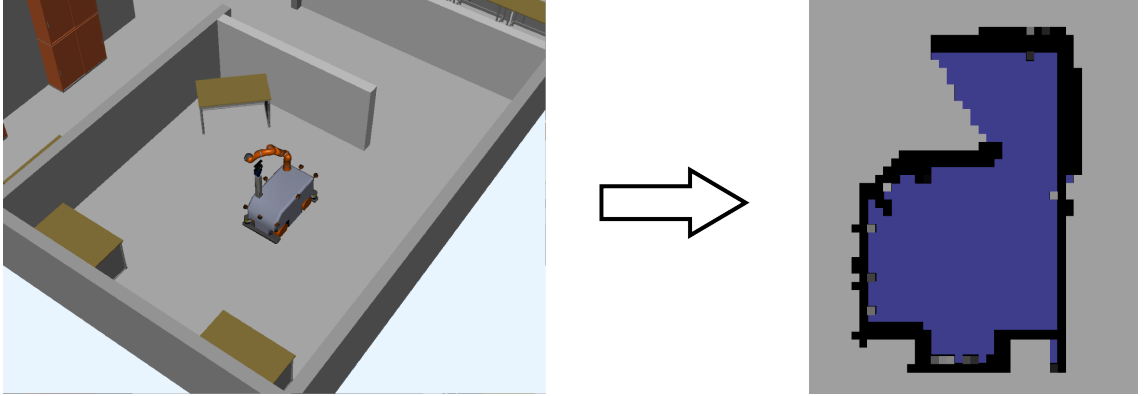


Figure 4.11: The environment of the simulated omniRob and the resulting voxel space in top-view, created by the SlamICP process.

is generated to a new pose, in order to gather new relevant information for the mapping process. This path must be free of any collision with the environment.

To give a better understanding of the respective algorithm steps, the whole process is explained by the help of one example. In Figure 4.11 on the right side, the voxel space on which the exploration process is based on is shown. On the left side in the picture, the simulated robot can be seen, standing in a small room with some tables. After several recordings of the environment, the SlamICP instance generates the depicted voxel map. The chosen grid edge length is  $l_{res} = 20\text{ cm}$ , which is definitely too imprecise for further application, but is better suited for the sake of demonstration.

For the exploration process, the following steps are performed consecutively:

1. Projection to Ground
2. Collision Space Generation
3. Frontier Generation
4. Application of exploration Strategy
5. Path Generation

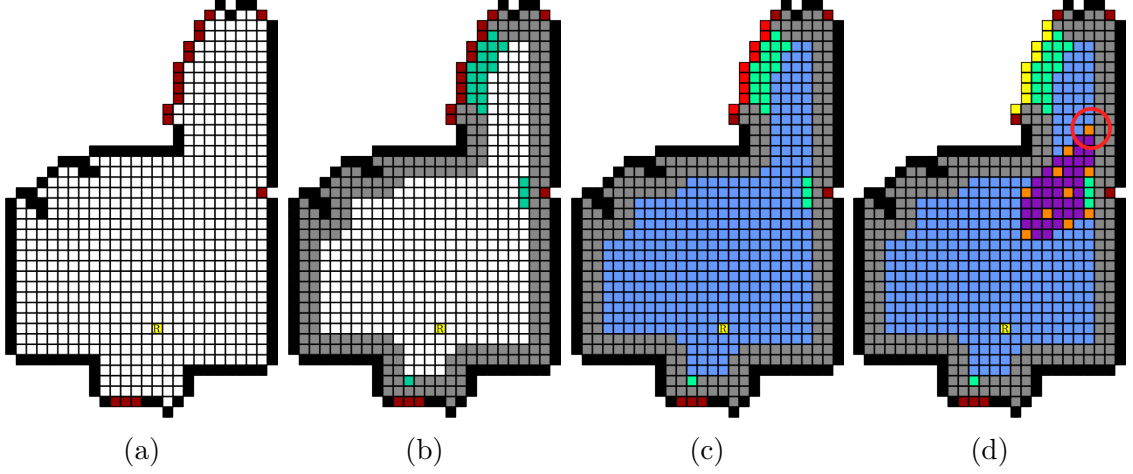


Figure 4.12: Visualization of the exploration process steps. The yellow pixel with the small "R" in it is the same in every picture and indicates the current location of the robot.

In the following, these steps are described and visualized in more detail.

#### 4.5.1 Projection to Ground

The robot is designed for indoor environments and in this thesis only limited to movements on a plane. Therefore, in order to make the voxel space  $\mathbf{V}$  more processable, it is projected onto a plane where the new 2D grid map is denoted as  $\mathbf{G}$ . This simplifies the distinction of obstacle free space and collision space without any relevant loss of information. This statement only holds true for box-like robots with no overhanging objects which may collide.

The 2D grid map  $\mathbf{G}$  is realized internally as a two-dimensional C++ `std::vector`, where each vector element includes a further `std::vector`. The cell value  $\mathbf{G}(i, j)$  describes the state of a specific cell. The edge length  $l_{res}$  of the 2D grid  $\mathbf{G}$  is same as for the 3D grid  $\mathbf{V}$ . The grid indices  $i$  and  $j$  describe the position for the  $x$ - or  $y$ -axis, respectively, and enable direct access for each cell state. Because of the vector implementation, only unsigned integer values are possible for the indices. This would make it impossible to represent the space at negative  $x$ - or  $y$ -axis positions. As a solution, an internal offset value is introduced which serves as a movement of the whole grid map to the minimal  $x$  and  $y$  position. This makes it possible to access the complete space without the need of implementing a new data structure and enables the use of negative  $i$  and  $j$  index values. Starting from a robot position  $\mathbf{r} = (x, y)^T \in \mathbb{R}^{2 \times 1}$  the resembling rasterized position  $\mathbf{s} = (i, j)^T \in \mathbb{Z}^{2 \times 1}$  is calculated by Equation 4.3.

$$i = \text{round}(x/l_{res}) \text{ and } j = \text{round}(y/l_{res}) \quad (4.3)$$

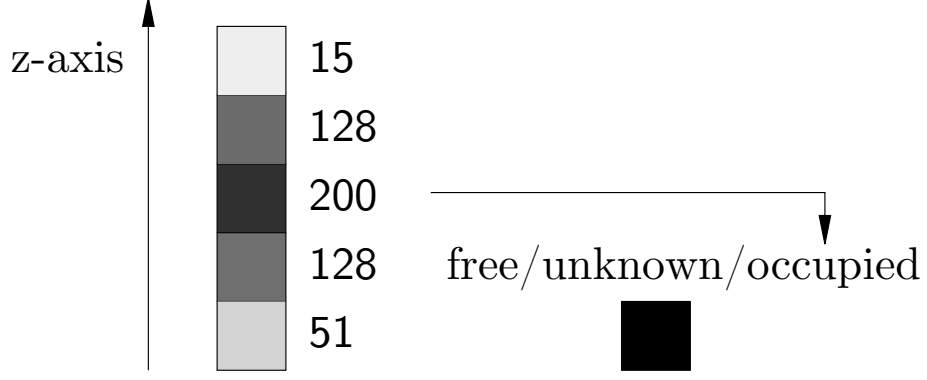


Figure 4.13: The process of iterating through a voxel column and projecting the correct state to the ground. The numbers indicate the occupancy value, where 0 or 255 resemble the occupancy probability values of 0 or 1, respectively.

The discretization causes a maximum distance error of  $e_1 = \frac{1}{\sqrt{2}}l_{res}$  (Equation 4.4). This is the distance from the middle point to the corner of a cell.

$$e_1 = \sqrt{\left(\frac{1}{2}l_{res}\right)^2 + \left(\frac{1}{2}l_{res}\right)^2} = \frac{1}{\sqrt{2}}l_{res} \quad (4.4)$$

In Figure 4.12a, the first processing step is visualized, where white defines *free*, black *occupied* and red *unknown* space.

Starting from the robot's position, the space  $\mathbf{V}$  is iteratively processed, in order to determine the whole connected free space. The algorithm starts with the grid cell  $\mathbf{G}(i, j)$ , in which the robot's center is currently located. The term  $\mathbf{V}(i, j, k)$  describes the same rasterized robot position, but in the 3D occupancy grid map, with  $k$  as a  $z$ -axis index. Each  $z$ -axis column of voxel will be traversed from a minimum  $k_{min}$  to a maximum value  $k_{max}$ . A minimum value has to be defined in order to separate floor from free space and a maximum value is needed in order to ignore the space above the robot, which the cameras cannot perceive.

Furthermore, certain thresholds are necessary for defining *free*, *unknown* and *occupied* space.

The occupancy values of the grid map in L3D range from 0 to 255, where all values below a certain threshold  $th_{free}$  are considered free and all values above  $th_{occ}$  are considered occupied. 255 resembles a probability of 1 for a cell of being occupied. The initial voxel state is 128. The narrower the free of occupied thresholds are defined, the more certainty is necessary for their definition. This consequently requires more measurements. In case one single *occupied* voxel is located in a voxel column, the respective 2D grid pixel in  $\mathbf{G}$  is set to *occupied* as well. Whereas the 2D grid value will only be set to *free*, if all voxels in the column hold the *free* state. In all other cases, the grid value is set to *unknown*. If the grid value is free, the process will

---

**Algorithm 2** Projection to Ground Algorithm

---

```
1:  $V$  = 3D voxel space,  $G$  = 2D grid map
2: push starting position indices  $(i_{start}, j_{start})$  to the end of queue  $Q$ 
3: while  $Q \neq \text{empty}$  do
4:   take and remove  $(i, j)$  from the front of  $Q$ 
5:   if  $(i, j)$  already processed then
6:     continue with 3
7:   end if
8:   move through corresponding column in  $col = V(i, j, k_{min} : k_{max})$ 
9:   if  $\exists$  voxel  $v$  in  $col$  with  $v \geq th_{occ}$  then
10:     $G(i, j) \leftarrow \text{occupied}$ 
11:   else if  $\exists$  voxel  $v$  in  $col$  with  $v > th_{free}$  then
12:     $G(i, j) \leftarrow \text{unknown}$ 
13:   else
14:     $G(i, j) \leftarrow \text{free}$ 
15:    push  $(i + 1, j), (i, j + 1), (i - 1, j)$  and  $(i, j - 1)$  to back off  $Q$ 
16:   end if
17: end while
```

---

continue with all untreated neighboring columns until no more free space is found. This procedure is similar to a simple flood-fill algorithm in computer graphics. The whole projection to ground algorithm is formulated in pseudo-code in Algorithm 2.

### 4.5.2 Collision Space Generation

In the next step, the robot geometry is taken into account. The robot position in the used coordinate system is defined by the robot's center, as can be seen in Figure 4.14. But to describe the robot only by its pose is not enough for a reasonable exploration

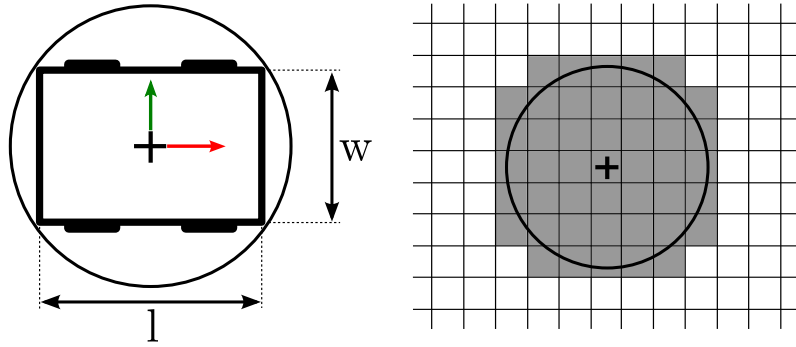


Figure 4.14: Robot geometry with the origin of the coordinate system indicated as a cross. In the grid map, the gray color indicates the cells, where no obstacle is allowed to be.

task. The dimensions of the robot have to be considered for every movement to prevent any collision with the environment. A certain distance has to be kept to every obstacle which has been perceived. Furthermore, this distance is also necessary towards regions which have not yet been perceived. The robot can not be sure if they will be free or occupied. In order to keep a certain minimum distance to any occupied or unknown space, a collision radius is used for the exploration task. It is applied to the grid map  $\mathbf{G}$  to determine all regions where the robot is allowed to go. The dimensions of the omniRob are described in subsection 4.1.1. The collision radius can be calculated with  $r_{coll} = \sqrt{(0.5l)^2 + (0.5w)^2}$  (Figure 4.14) and is applied to the grid map from the previous step. Every pixel, which is in reach of the collision radius of an *occupied* pixel is defined as collision space. The resolution error  $e_1$  has to be considered. This means, that every cell center, which is in reach of  $r_{coll} + \sqrt{2}l_{res}$  of an occupied or unknown cell, is defined as collision space. Around unknown pixels a *maybe collision* state is defined, because the robot can not yet be sure if there might be free or occupied space. This means, that it is not possible for the robot to go to this place without the chance of colliding with any object. The result is visualized in Figure 4.12b, where grey indicates *collision space* and green indicates *maybe collision space*. So far, the robot is able to rotate freely at every position, which is still considered free. The fact that the robot might be able to move closer to collision objects if it is rotated appropriately is not considered in the scope of this work.

### 4.5.3 Frontier Generation

In Figure 4.12c, the blue color defines all *reachable* space. In the depicted case, all previously declared *free* space is now *reachable* as well, but it might as well occur that some regions will be barred away by the collision space. A reachable border region from free to unknown space is called frontier.

During this step, the presence of a reachable border region is verified. The goal of the exploration task is to cover the whole environment, and therefore the robot has to find positions, where new data can be perceived. Therefore, all *maybe collision* space is evaluated for bordering unknown space. In the picture, this is visualized by a more bright green and red color. The grouping is performed by evaluating the neighborhood of one frontier cell. If two frontier cells share one edge or one corner, they belong to the same frontier. This is performed by iterating over all eight surrounding cells in the grid.

Normally, more than one of these regions will be present, and they are grouped together and sorted in ascending order, based on their calculated traveling distance. The processing of reachable space is performed by a Breadth-First Search (BFS) algorithm, and therefore it is easy to determine the grid distance to each border region.

#### 4.5.4 Application of Exploration Strategy

The next step is finding a new target pose for the robot in order to continue the exploration process. The implemented strategy is a frontier-based approach, where the information gain is considered. In case no appropriate target position can be determined, the exploration task stops and the environment is considered to be fully mapped. These three steps are performed consecutively.

1. Unknown Border Selection
2. Sample Space Generation
3. Scan Rating

They are explained more thoroughly in the following.

##### Unknown Border Selection

Depending on a certain strategy, one of the available frontiers is selected. Every frontier is a collection of grid cells  $\mathbf{G}(i, j)$ . The size of a frontier is determined by the number of grouped border cells. The number is also proportional to the selected grid size  $l_{res}$ . If several regions surpass a certain threshold size  $th_{f1}$ , the nearest one is chosen. The distance to a frontier is determined during the frontier generation step, where all reachable space is defined. This is to guarantee the continuous movement along a corridor and avoid a back and forth movement between several distinct frontiers.

In case no border region with the size above the first threshold can be detected, a second smaller threshold  $th_{f2}$  is considered. This may result from a fully explored room, in which only smaller passages, like a door are present and have not yet been explored in detail. If none is found at all, the environment is considered fully explored and the algorithm stops.

##### Sample Space Generation

Once a certain region has been selected, a sample space is created in front of it. This space marks possible positions from where the border can be observed. In Figure 4.12d, the selected border region is marked in yellow and the sample space in purple. A minimum distance to the unknown region is specified. This avoids new space to be in the robot's blind angle. In addition, a maximum distance is specified so as to limit the space of possible target positions.

Line rasterization is used, in order to determine the possibility of a direct line of sight towards the unknown border region. If the sample space was only determined by the distance to the frontier, there could be some space, e.g. behind a corner, where

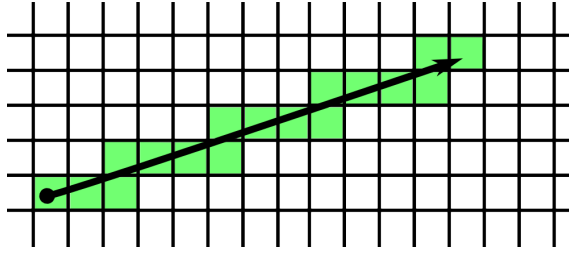


Figure 4.15: 4-connected Bresenham's line algorithm.

the frontier can not be seen by the robot. In order to avoid this, at least one frontier cell must be directly observable from every sample space cell. In Figure 4.15, the rasterization process is visualized. It is implemented as a 4-connected Bresenham algorithm and similar to 2D ray tracing. For every sample space pixel, at least some unknown border pixel have to be in direct line of sight, without the interception of an occupied pixel. This prevents the processing of positions, e.g. behind an edge, where the gain of new information is very unlikely or impossible.

In this generated sample space, several arbitrary robot positions are selected. The quantity is specified in a settings file and every sample point has a minimum distance to any other sample. The samples are visualized in Figure 4.12d as orange pixels. The rotation of the robot could be sampled as well, but in order to reduce the amount of necessary samples and the fact of an almost complete FoV (cf. Figure 4.2a) in front of the robot, its orientation is always directed towards the center of the unknown border region.

In case no sample space can be generated, this region is abandoned and the selection process returns to the *unknown border selection* step.

## Scan Rating

In the scan rating step, the sample with the most information gain is selected. This approach is conventionally used in NBV planning (cf. section 2.10) for object modeling. In the existing voxel space, a virtual measurement is performed. Similar to the integration of data into the space, the virtual measurement simulates beams for every camera pixel.

But instead of updating the space, the entropy is calculated without changing the voxel space  $\mathbf{V}$ . The information gain is calculated by the processing of all possibly observable occupancy grid cells. Further information is given in section 3.7. The sample pose with the highest information gain is likely to be the best one in the real environment and will therefore be the new target. In the example, this would be the top-right sample point, indicated by the red circle.

The implementation of the scan rating process takes a robot pose and generates

all beams for the ToF cameras at this pose with the maximum perception length. This process is very similar to the occupancy grid mapping step in subsection 4.4.3, where the beams for all cameras are generated, but based on the individual measurement. The 3D occupancy grid  $\mathbf{V}$  is now used to determine the information gain. In section 3.7, the process of calculating the information gain is described in more detail.

For the sake of performance, the amount of generated beams can be reduced by a factor. The camera sensor in this thesis has  $i \times j = 64 \times 48$  pixel values. For example, if a reduction factor of 3 is chosen, only every third beam in  $i$  and  $j$  direction will be processed. The overall evaluation steps are reduced by a factor of  $3 \times 3 = 9$ . This still results in an expressive information gain calculation with the advantage of a much higher processing speed. The scan rating step still demands a major amount of processing time of the complete autonomous exploration process.

### 4.5.5 Path Generation

An easy way of generating a path from a robot pose  $\mathbf{r}_i$  to another pose  $\mathbf{r}_{i+1}$  is using a search algorithm like BFS or A\* on the already generated grid map  $\mathbf{G}$ . The huge disadvantage of this approach is the high number of generated path segments. The robot always drives from one cell to another neighboring one. This can result in very abrupt and frequent changes in driving direction. As a consequence, the localization error is very likely to increase. Moreover, this approach will always lead to a path near the collision space around corners. This should be avoided to reduce the likelihood of accidental collisions. One solution to this problem in literature is the use of Voroni Diagrams [GMAM06], where the distance to any obstacle is maximized.

In this thesis, an already implemented path generation algorithm within the L3D library is used. It operates on the collision space grid map  $\mathbf{G}$  from Figure 4.12b. A starting position and a target position can be defined, where the generated path must completely lie within the collision free space. Additionally, obstacles are avoided and the distance to the collision space is maximized. In Figure 4.16, the path generation process is exemplified. The output of the path generation process step is an ordered list of robot positions. They define the path  $\mathbf{p} = \{(x_0, y_0)^T, (x_1, y_1)^T, \dots, (x_i, y_i)^T\}$  from start to end. The orientation of the robot is not changed during the path execution because of its omni-directional movement ability. Only at the final position, the robot is directed toward the center of the frontier, in order to exploit the denser FoV at its front-side.



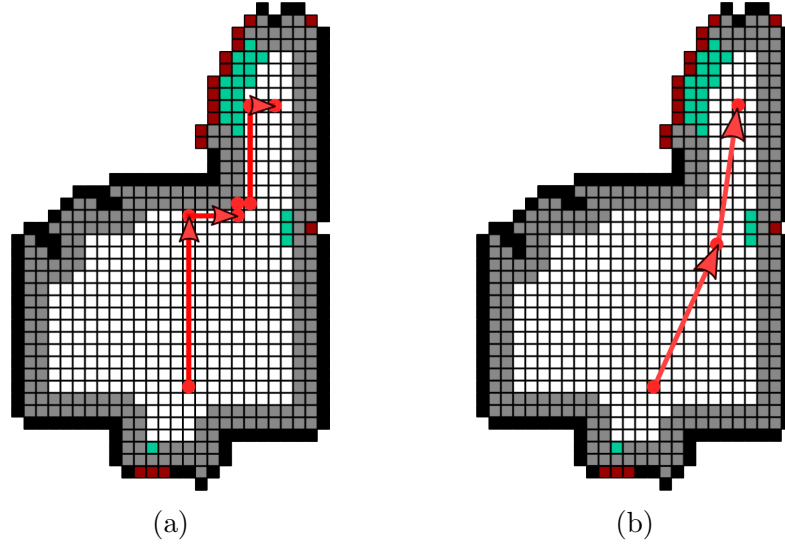


Figure 4.16: Path generation process. (a) would be a simple search strategy like BFS in the grid map, whereas (b) exemplifies the used implementation from the L3D library.

## 4.6 Motion Control

The robot's movement is prone to errors and noise. Therefore, the robot travels along the path in smaller steps. A maximum path segment length is defined, in order to keep the error manageable. After every segment, a pose correction by means of range scan alignment is performed. The environment is perceived with the ToF cameras and a point cloud is generated. As described in subsection 4.4.2, scan matching is performed to refine the position of the robot. In order to use the perceived information for further refinement of the map representation, the range data is integrated into the *metaview* and voxel space.

In Figure 4.4 the whole autonomous motion control process is depicted. After the exploration routine has generated a path to a new target observation position, the motion control process ensures the accurate reach of this position. After every path segment, it has to communicate with the SlamICP process (cf. section 4.4), where the data from the sensors is used for the pose refinement.

# Chapter 5

## Experiments and Discussion

In this chapter, the whole implemented autonomous exploration process is further evaluated and discussed in various aspects and the exploration abilities of the implemented process is demonstrated. First, one whole exploration process of a sample environment is presented. The test room is modeled in Blender according to the real test room at the DLR institute where the omniRob is currently standing. Afterwards, the results are discussed in matter of performance and behavior.

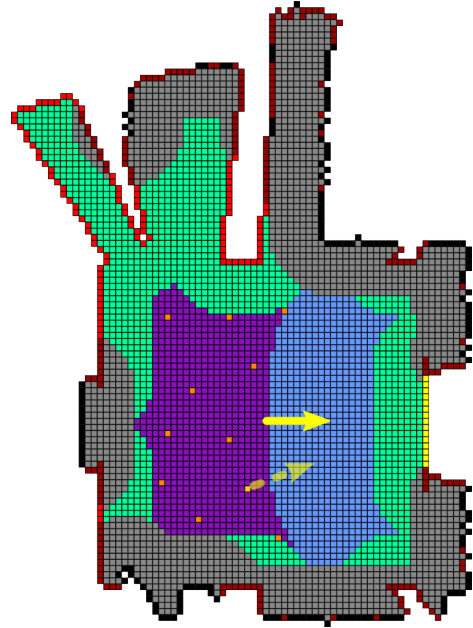
### 5.1 Example Exploration Process

In the following, an example autonomous exploration process will be presented and visualized. The size of the complete test environment is  $12\text{ m} \times 8\text{ m}$  and is modeled with Blender 2.72. The robot's starting position is visualized in Figure 5.1a. Its viewing direction is indicated by the yellow arrow. There are no narrow passages in the test environment. This makes it possible for the robot to drive everywhere and rotate freely within the safe collision space without colliding with any obstacle.

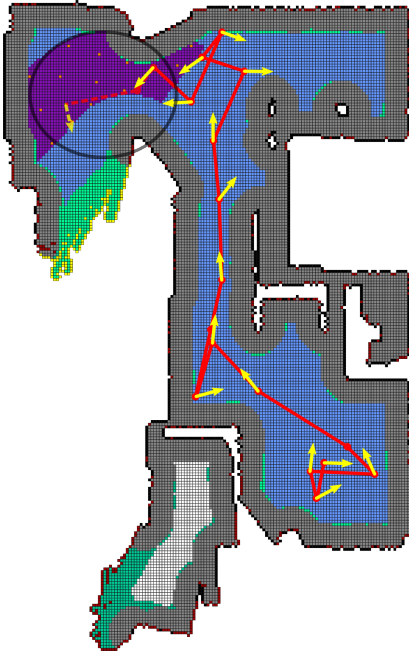
The grid resolution for the occupancy grid map is  $l_{res} = 5\text{ cm}$  and each cell obtains an occupancy value between 0 and 255. A value below  $th_{free} = 70$  is considered as free, whereas a value above  $th_{occ} = 240$  is considered occupied. The lower the threshold for free voxel is chosen, the more consecutive scans are necessary to define this space as free. 70 is a very loose threshold for the free space, but otherwise more environment recordings are necessary. The same applies for the occupied states. For the metascan, the distance to the nearest neighbor must be more than  $d_{min} = 5\text{ cm}$ . In the exploration process, 15 samples are evaluated at the maximum. Only the space between  $z_{min} = 10\text{ cm}$  and  $z_{max} = 50\text{ cm}$  height is considered for the space definition and exploration task. The space below 10 cm, is considered as floor, whereas above 50 cm, it is ignored completely. The FoVs of the cameras do not record this space in the current camera setting.



(a) MRE Simulator with the onmiRob starting position and rotation marked in yellow.



(b) The exploration grid map  $G$  after the first range measurement. The target pose is illustrated by the dotted arrow.



(c) The exploration map  $G$  after 14 exploration steps. The target pose is illustrated by the dotted lines.



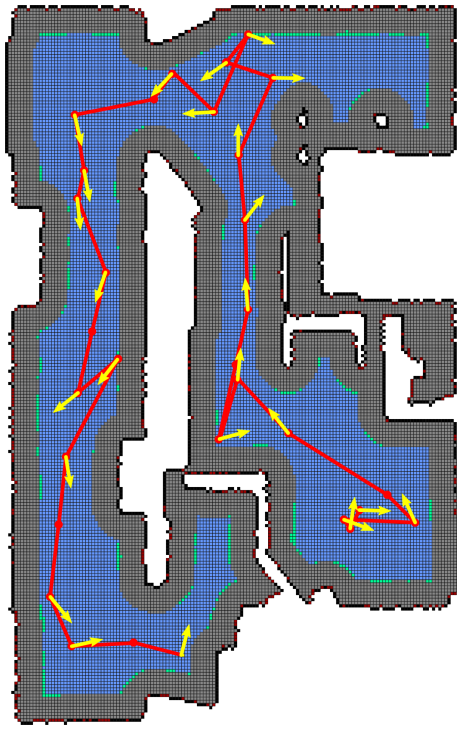
(d) The 3D occupancy grid map  $V$  after 14 exploration steps. The targeted frontier is marked by the red circle.

Figure 5.1: The exploration process in the beginning and during the autonomous execution, performed with the MRE simulator and a previously modeled test environment.

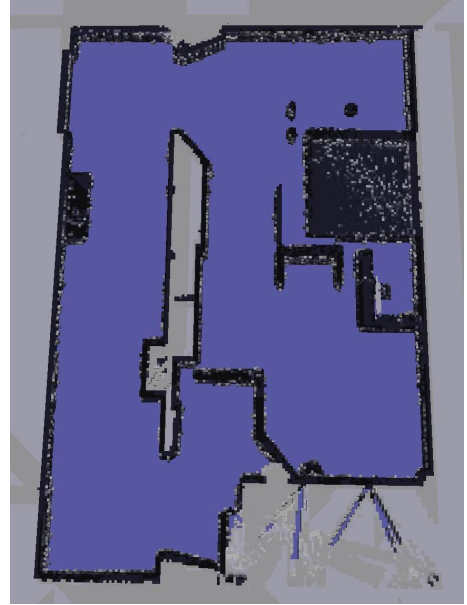
In Figure 5.1b, the implemented exploration map is depicted, after the first range measurement has been integrated into the occupancy grid map. The selected frontier is marked in yellow. Many parts of the surrounding environment are still missing, because the cameras do not cover the whole surrounding at once. This results from the application of several individual cameras with a limited angle of view. The environment can already be assumed by the black pixel, which represent the walls as occupied space. The nearest unknown border marked in yellow in front of the robot results from the limited angle of view of the front camera. Even after defining a space of  $2,5\text{ m} \times 2,5\text{ m}$  centered around the robot as free, some space near the ground cannot be perceived. All the other straight red pixel lines also represent the area, where the surrounding could not yet be perceived. But the robot is unrestricted enough to move around and register this space in the next step from a more distant point of view. The random samples, which are considered for the entropy-based evaluation are marked in orange. The dotted yellow arrow marks the selected next best robot pose out of the samples.

In Figure 5.1c, the exploration map is shown after the 14th step. The traveled path by the robot is visualized by the red lines. Every time the robot stops and takes a measurement is indicated by a red dot. The yellow arrows are the next best target poses after every execution of the exploration process. All red dots without a yellow arrow indicate a path execution, where a path is followed to a target position and only a relocalization is performed at this position. There exist robot positions very close to the collision space. In case of very inaccurate movements, this fact can cause severe problems, because the robot might collide with any obstacle. The preference of positions far away from any obstacle is not yet implemented within the scope of this thesis. The corresponding 3D occupancy grid map is visible at Figure 5.1d. The darker a voxel is depicted, the higher is its occupancy probability. The next target pose is indicated by the dotted arrow and line in Figure 5.1c and the frontier in the 3D occupancy grid map is highlighted by a red circle in Figure 5.1d.

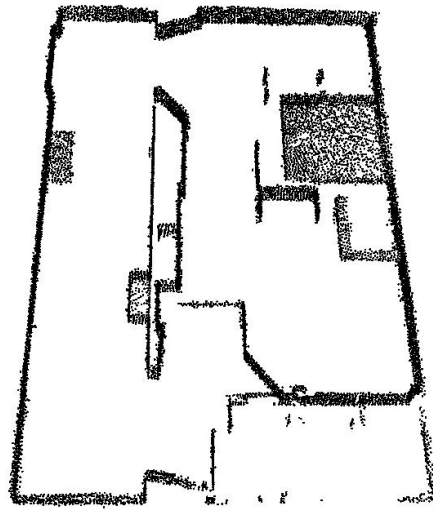
The result of the completed exploration process is shown in Figure 5.2. In Figure 5.2a, no new frontier could be determined by the implemented algorithm, thus the exploration stops. The entire path traveled by the robot is represented by the red lines and the respective generated next best positions are visualized by yellow arrows. The complete 3D occupancy grid map  $\mathbf{V}$  is depicted in Figure 5.2b. The metascan  $\mathbf{M}$ , which is needed for the ICP range scan alignment is pictured in Figure 5.2c and consists of 13 977 points. In case no point cloud reduction was performed and all point clouds were solely combined, the same exploration task would result in a point cloud with 545 133 individual points. There are still individual unknown pixels in between the occupied ones and they restrict the reachable area by the maybe collision space (indicated in green). But in order to certainly define them as either free or occupied, many more measurements would have to be taken. The sensor noise further hampers the exact definition of cell states. If an obstacle in the real world exactly borders the edge of an internal occupancy grid cell, it will take many measurements for the cell to diverge into one state. Therefore, these



(a) Exploration grid map  $G$  after the finished exploration process.



(b) 3D occupancy grid map  $V$  after the finished exploration process.



(c) The metascan  $M$  after the finished exploration process.

Figure 5.2: The completed exploration process and the resulting occupancy grid map and metascan.

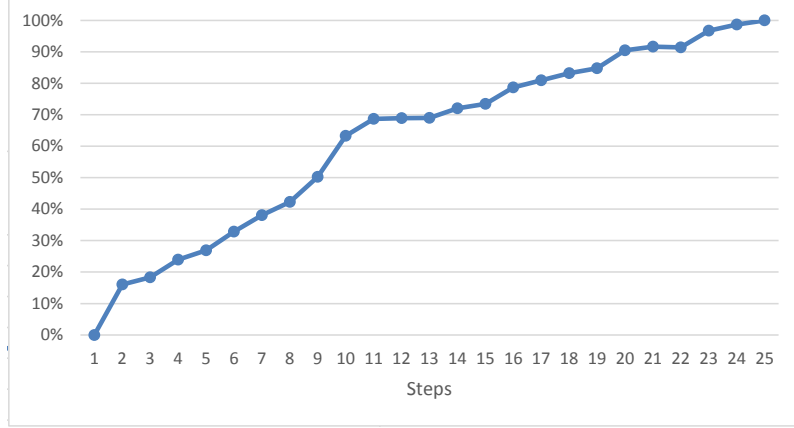


Figure 5.3: Environment coverage after every exploration step.

unreachable pixels are not considered anymore for the further exploration process.

In Figure 5.3, the environment coverage after each exploration step is depicted. The number of grid cells, which are defined as free, serve as an indicator for the map coverage. Initially, nothing is known about the environment. With the first measurement, the largest increase of map coverage is archived. This results from the complete surrounding of unknown space. The precondition before starting the autonomous exploration process is a safety distance to any obstacle in the environment. Therefore, the robot registers a lot of free space around itself, which results in the increased information gain. All preceding steps are iterative measurements in which the robot always stays within the known space. Hence, the gain of information will not be as large as in the first step. As soon as no new explorable frontier is present, the algorithm stops and the map is considered fully mapped.

## 5.2 Exploration Performance

In the previous section, an example exploration process is displayed and analyzed. In the following, the individual autonomous exploration process tasks is discussed in terms of processing time and performance. Exact times are difficult to determine because more than 20 threads running in parallel in the simulator and the individual times for the same task vary a lot. But it is still possible to give approximated average times for the same task in order to determine a qualitative characteristic of each processing step. The computer on which the simulation is performed possesses an Intel Xeon Processor W3520 with 2.66 GHz and 4 GB of RAM.

The most important factor for the performance of the implemented algorithm is the resolution of the occupancy grid. For the 3D occupancy grid  $\mathbf{V}$  and the projected 2D grid map  $\mathbf{G}$ , the edge length  $l_{res}$  is the same and highly influences the needed processing time. Another factor is the minimal distance  $d_{min}$  to any other point in

the metascan reduction step. The reduced metascan influences the processing time of the ICP algorithm and also the claimed RAM use. The SlamICP process does not scale with the number of consecutive measurements. Only the RAM usage is increasing, depending on the growth of the 3D octree space  $\mathbf{V}$  and the metascan  $\mathbf{M}$ . The exploration process does scale with the size of the map. Starting from the robot position, every reachable grid cell is searched. As a result, the algorithm will need a longer execution time as more free grid elements are present. During the exploration tasks, the size of the grid map is constantly growing. Also the frontier generation and robot pose sampling depends on the number of grid cells, which have to be processed.

The influence of the edge length  $l_{res}$  and the minimal distance  $d_{min}$  for the metascan are discussed in the following. The major CPU-intensive and time consuming steps performed in the SlamICP process are:

1. Generate point cloud  $\mathbf{P}$  from measurement [const].
2. Range scan alignment [ $d_{min}$ ].
3. Integration of measurement into metascan  $\mathbf{M}$  [ $d_{min}$ ].
4. Integration of measurement into 3D occupancy grid map  $\mathbf{V}$  [ $l_{res}$ ].

The parameter, on which the individual task depends on is noted in the square brackets [...]. For the exploration process, all tasks depend on the parameter  $l_{res}$ . These tasks are:

1. Projection to ground.
2. Collision space generation.
3. Frontier generation.
4. Robot pose sampling.
5. Scan rating.
6. Path generation.

The task of performing the measurement in the simulator with the eight cameras and accessing the images takes about 2 seconds within the MRE simulator. The path execution is not considered, because it highly depends on the robot's acceleration and speed parameters. The *generate point cloud  $\mathbf{P}$  from measurement* step in which the eight range images are converted into the point cloud  $\mathbf{P}$  takes 15 ms and remains constant. It is independent of the algorithm's parameters.

Reduction radius $d_{min}$	0	5	10	20	30	40	50	100	150
Integration into $\mathbf{M}$ [ms]	4.4	79	92	90	76	73	67	55	47
ICP alignment [ms]	886	848	784	254	169	65	40	12	10
Numer of points in $\mathbf{M}$	53238	50252	38481	16681	9500	5842	3995	1221	636

Table 5.1: Effect of  $d_{min}$  on the execution time and the metascan size.

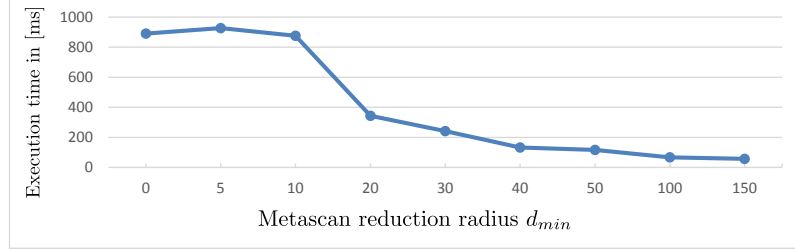


Figure 5.4: The execution time of the metascan point reduction and the ICP range scan alignment conditioned on the parameter  $d_{min}$ .

### 5.2.1 Point Cloud Reduction

In Table 5.1, the processing times for the two tasks *Range scan alignment* and *Integration of measurement into metascan  $\mathbf{M}$*  are listed, conditioned on the parameter  $d_{min}$ . For each test, the same sequence of 10 range measurements and odometry inputs was used in order to ensure a reasonable comparison. If no reduction radius is chosen, that means  $d_{min} = 0$ , no reduction is performed and the integration of one measurement into the metascan only lasts about 4.4 ms. The ICP operates on the full dense point cloud and therefore needs the most time. In case the point reduction is performed, the size of the metascan decreases a lot, which results in a faster execution of the ICP alignment. The reduction itself also needs time, because the near vicinity of a new point has to be searched in a space octree structure. The effect of the parameter  $d_{min}$  on the execution time is depicted in Figure 5.4, in which the time of the metascan integration and of the range scan alignment are combined.

Naturally, a high reduction of the metascan will result in a worse representation of the environment and increase the alignment error. For the modeling of smaller objects like a table leg, a reduction radius of more than 50 mm is not suitable. The best results for the algorithm and the test environment in this thesis were archived with a reduction radius of 20 mm. The environment can still be adequately modeled with a high accuracy and a reduced execution time. Furthermore, the RAM usage is not directly proportional to the number of measurements, like in case of no reduction, but only to the size of the environment.



Grid resolution $l_{res}$ [mm]	10	20	30	40	50	60	80	100	120
Integration into $\mathbf{V}$ [ms]	3308	1767	1238	892	737	592	456	423	319
Projection to ground [ms]	1085	214	84	38	20	15	8.1	6.0	2.4
Collision space generation [ms]	84	12	3.9	1.7	0.9	0.8	0.4	0.2	0.1
Frontier generation [ms]	29	8.7	3.5	2.0	1.4	0.9	0.6	0.4	0.2
Robot pose sampling [ms]	3240	201	48	34	11	6	1.8	1.2	0.8
Scan rating (single sample) [ms]	1124	637	458	359	285	236	172	139	119
Path generation [ms]	15388	1660	941	737	674	622	591	589	590

Table 5.2: The execution times of all process steps, which depend on the grid resolution  $l_{res}$ .

### 5.2.2 Grid Resolution

The grid resolution has a lot of influence on the execution time and accuracy of the presented algorithm. As a first obvious consequence, the environment can be represented more accurately, if a smaller grid resolution is selected. But this parameter can not be decreased without any limit. The processing time and RAM usage will strongly increase for smaller resolutions. In order to get an impression about the individual processing times, a wide range of grid resolutions has been tested. For the execution times in Table 5.2, an exploration run was performed in the same test environment with the same amount of exploration steps. The displayed numbers are mean values of the needed execution time. Some tasks also scale with the size of the environment, but as it is growing to almost the same size for every used grid resolution, this increase does not eliminate the comparability. The test environment displayed in Figure 5.1a was used. The integration of the measurement into the 3D occupancy grid  $\mathbf{V}$  does belong to the SlamICP process and does not scale with the map size. For the scan rating step, the execution time for one single sample is displayed. This time scales linear with the number of pose samples. If more than 10 samples are evaluated, it is the main factor for the whole exploration task. In Figure 5.5, the execution times of the exploration task are displayed in relation to their partial share and with the evaluation of 10 samples for each processing step.

It is clearly visible, that the scan rating step requires the most time in relation to the other steps. In the path generation step, an existing L3D class is used. First, the own internal map representation has to be converted into the appropriate format, then a graph is created on which the path from one to the other position is generated. This step also utilizes a lot of execution time. Furthermore, in contrast to the scan rating step, the path generation step scales with the size of the map. In order to reduce the performance of the whole exploration process, the path planing on a 2D grid map has to be optimized. The entropy based scan rating process for several samples takes the major part of the whole process. To reduce this processing time, the evaluation could be performed on the 2D grid map, or other reasonable analytic strategies have to be found. The other process steps are negligible in processing time in the current approach.

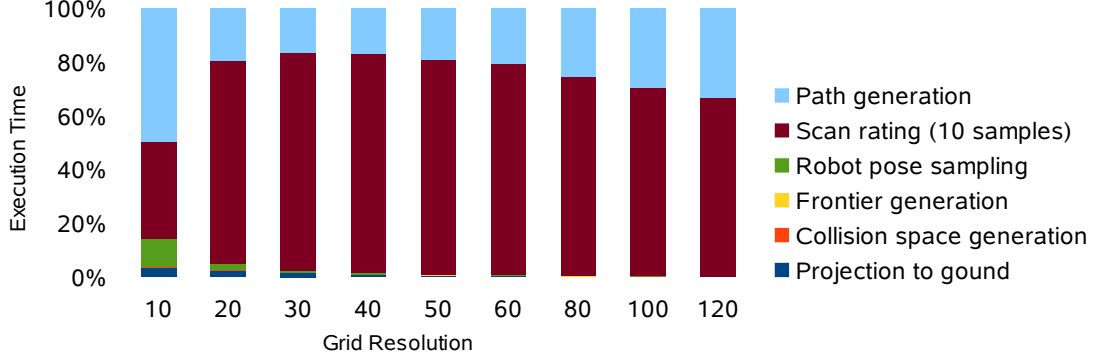


Figure 5.5: The partial share in execution time of each step in the exploration process conditioned on the grid resolution  $l_{res}$ . (The *integration into  $V$*  step belongs to the SlamICP process and is not included in the diagram.)

It has to be stated, that a grid resolution of  $l_{res} = 10$  mm is not applicable anymore for the underlying problem setting. The 4 GB RAM are not sufficient anymore after several measurement steps. The processing time highly increases and a visualization of the map representation is not possible anymore with the used computer. Furthermore, the integration of the individual beams into the 3D occupancy grid map does not deliver useful results. The individual grid cells, which intersect with the generated beams from the depth measurement do not lie side by side anymore, but leave some space unprocessed between them. This effect occurs some distance away from the camera till to the end of the measuring distance and the result is visualized in Figure 5.6. A multitude of measurements would be necessary to cover the whole free space.

A solution to this problem would be a widening of the individual beams linear to their distance, or the use of cameras with a higher resolution. The best exploration results for the test environment were archived with a grid resolution of  $l_{res} = 50$  mm to  $l_{res} = 80$  mm and a point cloud reduction of  $d_{min} = 30$  mm.

## 5.3 Major Drawbacks

### 5.3.1 Metascan Misalignment

The main disadvantage of the presented algorithm in this thesis is the missing correction and relaxation of previously integrated measurements. One range measurement consists of the combination of eight distinct range measurements into one point cloud. The individual measurements of each camera do not necessarily overlap, but contain free space between them. The metascan is an accumulation of the resulting

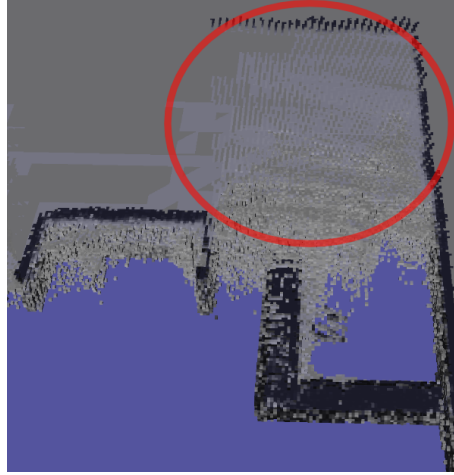
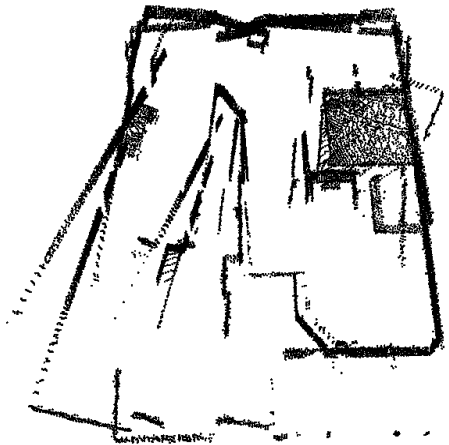
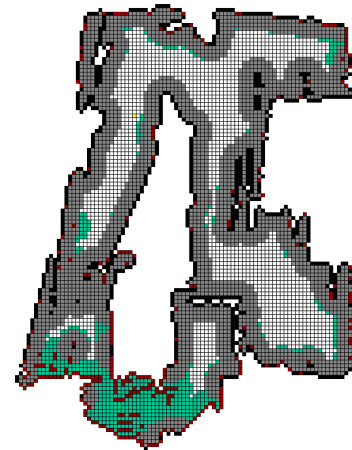


Figure 5.6: Occupancy grid map after range scan integration. A grid resolution of 10 mm is too low for a reasonable voxel mapping.



(a) Metascan  $M$  misalignment during the exploration.



(b) Resulting exploration map  $G$ .

Figure 5.7: Incorrect exploration caused by metascan misalignment. The resulting exploration map does not represent the real world environment.

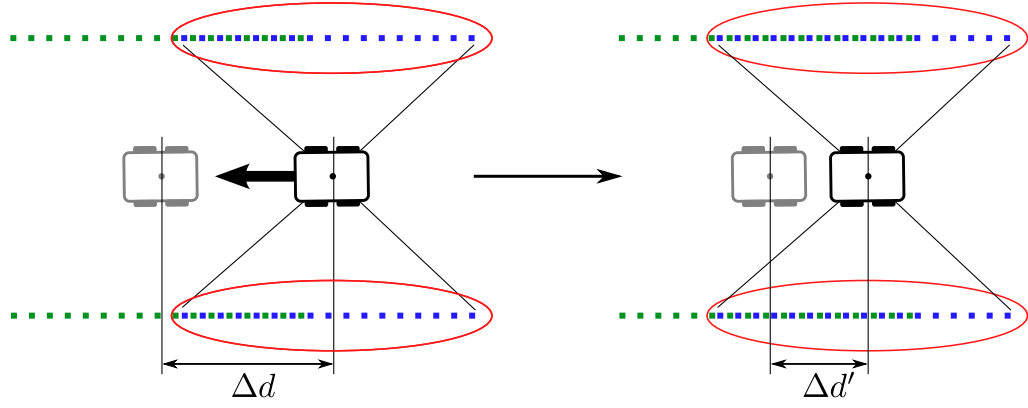


Figure 5.8: Shortening of long pathways without shape indicators by range scan alignment.

point clouds and can contain some missing space as well. If the odometry estimate of a new robot pose is not very exact and the correspondence search of the range scan alignment selects the wrong points, the resulting transformation will lead to a wrong pose refinement. The gathered data is incorrectly integrated into the existing metascan. All further range scan alignments will take this faulty metascan as a map reference. The algorithm is not able to correct the fault and will diverge and cancel. An exemplary misalignment is depicted in Figure 5.7, in which the faulty metascan and the resulting exploration map are visible. Some few alignment errors resulted in an erroneous environment representation. Ideally, such an alignment error only results in a premature ending of the exploration process. But for the path execution, a severe alignment error can even lead to collisions with the environment, in case the localization after each path segment is completely wrong.

In Figure 5.8, another alignment problem is displayed, caused by the shape of the environment itself. This especially happens in long unchanging pathways without distinct shape features. In case the robot is moving in a long pathway, where only straight walls are present and no standing around objects influence the shape of the range scan, this pathway will be shortened by the ICP alignment. This situation is visualized in Figure 5.8, in which a robot registers the environment from a new pose to the right side. But because of the unchanging environment along the pathway, the new range scan is identical to the previous one. The range scan alignment will match the two point clouds and correct the robot pose to the left. The exact maximum length of the path shortening is defined by the correspondence search radius for the ICP algorithm. A smaller radius will reduce the shortened length. But for very inaccurate odometry sensors, a small search radius might lead to failing or wrong alignments. Within the scope of this thesis the underlying problem of missing shape indicators in the environment is not considered explicitly. The targeted indoor environments have many obstacles and corners in which a path shortening is of no concern.

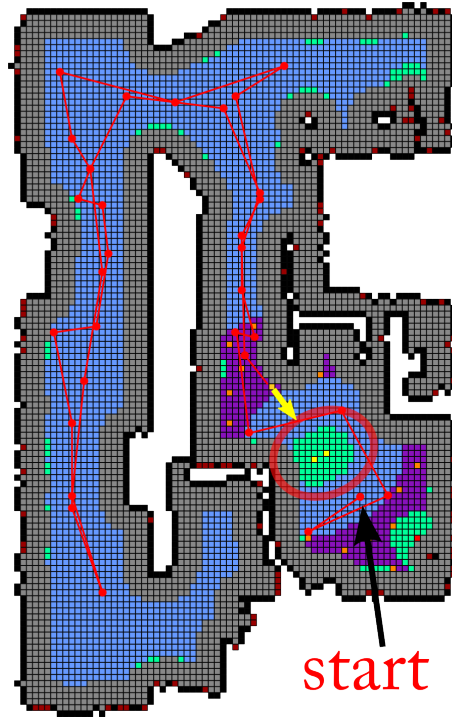


Figure 5.9: Some unknown space at the starting position causes the robot to drive back the whole way. This could be prevented by a map segmentation.

### 5.3.2 Missing Map Segmentation

One other drawback of the implemented algorithm is the missing map segmentation. In case the robot is exploring an unknown environment, it continues to follow the nearest frontier to unknown space. Sometimes it happens, that there is still unknown space present in a room, when the robot already continues to explore the region in the corridor or the next room. When everything else of the environment is already covered, the robot will move back to the partly explored room. This problem is illustrated in Figure 5.9, where some small space is still unexplored near the starting position. The few unexplored cells marked in yellow, which were not fully covered by one of the ToF cameras, cause the robot to drive back the whole way. In case the robot intends to fully explore a room or a bigger connected area, before moving to a next room, this problem could be avoided. Therefore, the map must be segmented into different parts.

Furthermore, this segmentation would have positive effects on the processing time of the algorithm. If a room is fully explored, it can be treated finalized, and does not have to be considered for the exploration process algorithms anymore. Also for future work, if the existing map representation needs to be corrected, a map segmentation can highly increase the performance.

### 5.3.3 Noisy Measurement

The implemented algorithm is very sensitive to sensor noise. The robot only moves to a place if every voxel column in the 3D occupancy grid map is defined as free within the collision radius. If one of the cells is not below a predefined occupancy threshold, the robot is not allowed to travel there. Especially for the separation of floor and obstacles, the sensor noise has a hampering effect. In the scope of this thesis, the raw point cloud is used for the environment modeling. In order to determine the traversable and reachable environment, only the space above a certain threshold is considered. For the noise parameters of the cameras from the manufacturer, a threshold of 10 cm is used in this thesis. If a lower threshold is selected, no connected free space can be defined, because the sensor noise will lead to many grid cells above the floor to be either unknown or occupied. The exact process of the projection algorithm is described in subsection 4.5.1. This also means that no object below a height of 10 cm will be recognized.

Smaller objects like vertical table legs can not be registered and integrated into the exploration process. If such small objects are present in the targeted environment, the robot is likely to ignore them. This can result in damage of the respective object, or the robot itself. In addition, the odometry and camera noise can encourage an error in the range scan alignment, as described in subsection 5.3.1. All in all, a low sensor noise with few outliers is crucial for a reliable exploration of an unknown environment. Otherwise, an early abortion of the exploration process or even a collision with an obstacle are very certain.

# Chapter 6

## Conclusion

### 6.1 Summary

The implemented autonomous exploration approach in this thesis is a combination of frontier- and entropy-based exploration. For the mapping and localization process, a metaview registration SLAM algorithm has been implemented, which operates on the raw range measurements of depth cameras. The individual measurements are combined to a metascan and integrated into a 3D occupancy grid map. In order to reduce the amount of necessary RAM, a point reduction method is implemented for the integration of new data into the metascan. Furthermore, a 3D occupancy grid map is generated by the SlamICP process. The occupancy grid map is realized by a memory saving octree data structure, which is available in the L3D C++ library. The exploration process is based on the generated 3D occupancy grid map. Because of the robot movement, which is restricted to planar environments, this 3D representation is projected to a 2D grid map, in which the collision spaces and available frontiers are determined. The frontier selection process is determined analytically, whereas the robot pose sampling is randomized. The evaluation of each sample is performed by an entropy-based scan rating process within the 3D occupancy grid map. For a faster execution of the scan rating process, the amount of generated test beams is reduced by a predefined factor. The path generation from one to another robot pose is performed by an existing L3D C++ class.

An example autonomous exploration task was described in section 5.1 and the performance of the whole process was analyzed in section 5.2. The effects of the grid resolution  $l_{res}$  and the metascan point reduction radius  $d_{min}$  were evaluated in detail and the major time consuming tasks were detected. The algorithm was tested within the MRE simulator. An integration and testing of the algorithm with the real robot could not be performed, because of different version conflicts and interface problems. But the simulation results are expected to be applicable for the real world scenario. It showed that the scan rating and path generation steps require

the longest execution time, whereas the 2D grid projection and processing are almost negligible. The implemented SlamICP process does not scale with the map size and iteration steps, but no faulty alignments and noise errors can be corrected. For smaller environments and sensors with low noise, this approach is still applicable. The implemented algorithm supplies a testable and working framework for further developments. The interfaces of each process are clearly defined and enable the extension to further problem solutions, e.g. the detection of negative slants, descending stairways or the correction of erroneous alignments.

## 6.2 Future Work

The presented approach in this thesis is similar to the RBPF approach with only one single particle. As a consequence, no loop closing for large-scale environments is possible. This makes the implemented approach only useful for smaller indoor environments. Furthermore, individual severe alignment errors will cause erroneous results.

There are basically two ways of solving this problem and enhance the current approach for large-scale environments. The first approach is the development of a performant RBPF algorithm for the 3D case. As shown in [WSBC10], involved data structures need to be implemented first, to ensure an applicable SLAM routine with enough individual particles. The available RAM and the processing power are the crucial factors for this strategy. If only a few particles are used, this approach will also be very likely to diverge after a certain time.

Another solution is the integration of graph construction and relaxation algorithms - in literature often referred to as Front-End and Back-End. The individual robot poses can be interconnected to a graph, in which the spacial constraints between the individual poses form an optimization problem. The drawback of this approach is the steadily increasing amount of necessary processing power. After every relaxation of the graph, the occupancy grid map would have to be rebuilt. Furthermore, the nodes in the graph are constantly increasing in number after each iteration step, which results in a more complex optimization problem. A solution to this issue could be the segmentation of the environment into delimited areas (e.g. rooms), which will only be changed if necessary. Otherwise, only the individual areas are relaxed in the optimization problem. Furthermore, a segmentation of individual rooms would be of high interest for the indoor exploration task. As soon as one room is fully explored, it does not have to be considered anymore for further processing.

For the exploration process itself, it is also beneficial to target robot poses, where a correct loop closing is very likely. To visit already observed places again increases the probability of a proper alignment for the graph-based approach and supports the propagation of correct particles in the RBPF approach. In order to reduce the



processing data of the whole algorithm, a feature and landmark extraction procedure can be applied to the sensor data. Geometric shapes can be extracted from a point cloud and replace the large amount of individual points. Especially for indoor environments, the detection of walls and doors provide a highly reduced environment representation.

Another interesting problem in the autonomous exploration research is the consideration of the robot's geometry. So far, the distance from the robot to any object in the environment is determined large enough to prevent any collision, regardless of the robot's rotation. But especially for narrow corridors or passages, the robot might only fit through if it is rotated correctly. The robot also could have a different shape than the box-like robot in this thesis. The reasonable consideration of these factors into the autonomous exploration task can extend the applicability and performance for more restrained and narrow environments.

A further interesting consideration is the use of the two 2D horizontal range sensors, which are already mounted on the omniRob, for the SLAM process. They are attached on a low height near the floor and enable a better recognition of small obstacles. Furthermore, they provide a complete panoramic perception of the environment. KUKA already supplies the omniRob with a RBPF-based SLAM algorithm, which is based on the 2D sensors. The 3D data from the ToF cameras can then be used additionally for the exploration process in order to model overhanging obstacles or descending slants. The generation of the 3D representation can then be performed after the exploration process. This so called *offline processing* moves a lot of time consuming calculations to a post-processing step and provides more performance for motion control and path planning algorithms.

In this thesis, the exploration process is performed in a stop-and-go fashion. The robot moves to a target position, stops, takes a recording, integrates the perceived data into the environment model and continues its movement. For the fast execution of the whole exploration task, a subdivision of the whole process into different threads would be beneficial. The localization and mapping tasks can be performed in a different thread than the robot movement or motion control task. This would enable the possibility of an environment exploration and mapping algorithm, without the need of stopping the robot movement at every path segment. The implementation of smooth trajectories for the path planning and the concurrent execution of the robot movement with environment mapping are only a few out of many other enhancement possibilities.

The development and functionality of an autonomous exploration process is presented and analyzed in this thesis. It states a foundation for many further developments and possible extensions. The topic of autonomous exploration including SLAM, path planning and motion control covers a very large field of research. As shown, the development of many interesting and involved algorithms is imaginable. To prove and test their applicability for real world scenarios with the underlying hardware setting is up to subsequent researchers.

# List of Figures

1.1	The different parts of robotic exploration. . . . .	3
2.1	Frontier-based exploration. . . . .	10
2.3	Next Best View exploration process. . . . .	14
3.1	Online and Full SLAM Dynamic Bayesian Network (DBN) . . . . .	16
3.2	Robot pose in 2D coordinate frame. . . . .	17
3.3	Working principle of range sensors. . . . .	18
3.4	Exemplary ToF camera range measurement. . . . .	19
3.5	Metaview registration with the ICP algorithm. . . . .	20
3.6	Octree data structure and an update step for the occupancy grid map. . . . .	24
3.7	The occupancy grid entropy for a single cell. . . . .	28
4.1	The used omniRob platform and ToF cameras. . . . .	31
4.2	OmniRob FoV visualization. . . . .	32
4.3	The omniRob in the MRE simulator with the FoV of all cameras visualized. . . . .	34
4.4	Autonomous exploration process workflow. . . . .	35
4.5	The encapsulated SlamICP process. . . . .	36
4.6	Point cloud generation. . . . .	37
4.7	ICP range scan alignment of two consecutive data sets. . . . .	38
4.8	Metascan for a test environment. . . . .	39
4.9	3D occupancy grid map with a resolution of 5 cm edge length. . . . .	40
4.10	The interfaces of the exploration process. . . . .	41
4.11	The simulated omniRob in MRE and the resulting voxel space. . . . .	41
4.12	Visualization of the exploration process steps. . . . .	42
4.13	The process of iterating through a voxel column and projecting the correct state to the ground. . . . .	43
4.14	Robot geometry with the origin of the coordinate system indicated as a cross. . . . .	44
4.15	4-connected Bresenham's line algorithm. . . . .	47
4.16	Path generation process. . . . .	49
5.1	The exploration process in the beginning and during the autonomous execution. . . . .	51

5.2	The completed exploration process and the resulting occupancy grid map and metascan. . . . .	53
5.3	Environment coverage after every exploration step. . . . .	54
5.4	The execution time of the metascan point reduction and the ICP range scan alignment conditioned on the parameter $d_{min}$ . . . . .	56
5.5	The partial share in execution time of each step in the exploration process conditioned on the grid resolution $l_{res}$ . . . . .	58
5.6	Occupancy grid map after range scan integration. . . . .	59
5.7	Incorrect exploration caused by metascan misalignment. . . . .	59
5.8	Shortening of long pathways without shape indicators by range scan alignment. . . . .	60
5.9	Some unknown space at the starting position causes the robot to drive back the whole way. . . . .	61

## List of Tables

5.1	Effect of $d_{min}$ on the execution time and the metascan size. . . . .	56
5.2	The execution times of all process steps, which depend on the grid resolution $l_{res}$ . . . . .	57

# Bibliography

- [AC10] Francesco Amigoni and Vincenzo Caglioti. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems*, 58(5):684–699, 2010.
- [AF88] Nicholas Ayache and Olivier D Faugeras. Building, registering, and fusing noisy visual maps. *The International Journal of Robotics Research*, 7(6):45–65, 1988.
- [AG05] Francesco Amigoni and Alessandro Gallo. A multi-objective exploration strategy for mobile robots. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18–22, 2005, Barcelona, Spain*, pages 3850–3855, Barcelona, Spain, April 2005. IEEE.
- [BATP10] Nicola Basilico, Francesco Amigoni, Letizia Tanca, and Barbara Pernici. *Navigation Strategies for Exploration and Patrolling with Autonomous Mobile Robots*. PhD thesis, Politecnico di Milano, 2010.
- [BDW06] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine*, 13(3):108–117, 2006.
- [BG10] Gabriel I. Barbash and Sherry A. Glied. New technology and health care costs - the case of robot-assisted surgery. *New England Journal of Medicine*, 363(8):701–704, 2010.
- [BM92] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *The IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, February 1992.
- [BMF<sup>+</sup>00] Wolfram Burgard, Mark Moors, Dieter Fox, Reid Simmons, and Sebastian Thrun. Collaborative multi-robot exploration. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24–28, 2000, San Francisco, CA, USA*, volume 1, pages 476–481, San Francisco, CA, USA, April 2000. IEEE.

- [BSG05] Wolfram Burgard, Cyrill Stachniss, and Giorgio Grisetti. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.
- [BZW<sup>+</sup>95] Joseph E. Banta, Yu Zhien, Xiao Zhao Wang, G. Zhang, T. Smith, M. and Mongi A. Abidi. A “best-next-view” algorithm for three-dimensional scene reconstruction using range images. In *Photonics East’95*, pages 418–429. International Society for Optics and Photonics, 1995.
- [Chv75] Vasek Chvatal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory, Series B*, 18(1):39–41, February 1975.
- [CM92] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.
- [DDFMR00] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 176–183. Morgan Kaufmann Publishers Inc., 2000.
- [DW88] Hugh F Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, 1988.
- [DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous Localization and Mapping (SLAM): Part I the essential algorithm. *Robotics & Automation Magazine*, 13(2):99–110, 2006.
- [DWRN96] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. Localization of autonomous guided vehicles. In *Robotics Research*, pages 613–625. Springer, 1996.
- [FM08] Stefan Fuchs and Stefan May. Calibration and registration for precise surface reconstruction with time-of-flight cameras. *International Journal of Intelligent Systems Technologies and Applications*, 5(3):274–284, 2008.
- [FO05] Luigi Freda and Giuseppe Oriolo. Frontier-based probabilistic strategies for sensor-based exploration. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, ICRA 2005, April 18–22, 2005, Barcelona, Spain*, pages 3881–3887, Barcelona, Spain, April 2005. IEEE.
- [GBL02] Hector H Gonzalez-Banos and Jean-Claude Latombe. Navigation strategies for exploring indoor environments. *The International Journal of Robotics Research*, 21(10-11):829–848, 2002.

- [GKSB10] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.
- [GMAM06] Santiago Garrido, Luis Moreno, Mohamed Abderrahim, and Fernando Martin. Path planning for mobile robot navigation using voronoi diagram and fast marching. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006*, pages 2376–2381, Beijing, China, October 9–15 2006. IEEE/RSJ, IEEE.
- [GN01] Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.
- [GOR94] Javier Gonzalez, Anibal Ollero, and Antonio Reina. Map building for a mobile robot equipped with a 2d laser rangefinder. In *Proceedings of the 1994 International Conference on Robotics and Automation, San Diego, CA, USA, May 1994*, pages 1904–1909, San Diego, CA, USA, May 1994. IEEE Computer Society.
- [GSB07] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.
- [GSKB11] G Grisetti, H Strasdat, K Konolige, and W Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9–13 May 2011*, Shanghai, China, May 2011. IEEE.
- [HBAB10] Dirk Holz, Nicola Basilico, Francesco Amigoni, and Sven Behnke. Evaluating the efficiency of frontier-based exploration strategies. In *Proceedings of the joint conference of the 41st International Symposium on Robotics (ISR 2010) and the 6th German Conference on Robotics (ROBOTIK 2010)*, pages 36–43, Munich, Germany, June 2010. VDE VERLAG.
- [HKH<sup>+</sup>10] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *The 12th International Symposium on Experimental Robotics (ISER)*. Citeseer, 2010.
- [JSPB07] Dominik Joho, Cyrill Stachniss, Patrick Pfaff, and Wolfram Burgard. Autonomous exploration for 3d map learning. In Karsten Berns and Tobias Luksch, editors, *Autonome Mobile Systeme 2007, 20. Fachgespräch, Kaiserslautern, 18./19. Oktober 2007*, Informatik Aktuell, pages 22–28, Kaiserslautern, Germany, October 2007. Springer.

- [Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.
- [KTH01] Sven Koenig, Craig Tovey, and William Halliburton. Greedy mapping of terrain. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation, ICRA 2001, May 21–26, 2001, Seoul, Korea*, volume 4, pages 3594–3599, Seoul, Korea, May 2001. IEEE.
- [LDW91] John J. Leonard and Hugh F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems '91, Intelligence for Mechanical Systems, Proceedings IROS'91*, pages 1442–1447. IEEE, 1991.
- [LM97] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.
- [ME85] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, Missouri, USA*, volume 2, pages 116–121, 1985.
- [MSW01] Stewart J. Moorehead, Reid Simmons, and William L. Whittaker. Autonomous exploration using multiple sources of information. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation, ICRA 2001, May 21–26, 2001, Seoul, Korea*, volume 3, pages 3098–3103, Seoul, Korea, May 2001. IEEE.
- [MTKW02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI/IAAI*, pages 593–598, 2002.
- [MTKW03] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003. IJCAI.
- [MTS07] Michael Montemerlo, Sebastian Thrun, and Bruno Siciliano. *Fast-SLAM: A scalable method for the simultaneous localization and mapping problem in robotics*, volume 27. Springer, 2007.
- [MWBDW02] Alexei A. Makarenko, Stefan B. Williams, Frederic Bourgault, and Hugh F. Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, September 30 - October 4, 2002*, vol-

- ume 1, pages 534–539, Lausanne, Switzerland, September/October 2002. IEEE.
- [Nüc09] Andreas Nüchter. *3D robotic mapping: the simultaneous localization and mapping problem with six degrees of freedom*, volume 52. Springer, 2009.
- [OVFT04] Giuseppe Oriolo, Marilena Vendittelli, Luigi Freda, and Giulio Troso. The SRT method: Randomized strategies for exploration. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, volume 5, pages 4688–4694, New Orleans, LA, USA, April 2004. IEEE.
- [Pit96] Richard Pito. A sensor-based solution to the “next best view” problem. In *Proceedings of the 13th International Conference on Pattern Recognition 1996*, volume 1, pages 941–945. IEEE, 1996.
- [Pul99] Kari Pulli. Multiview registration for large data sets. In *Second International Conference on 3-D Digital Imaging and Modeling, 1999. Proceedings*, pages 160–168. IEEE, 1999.
- [SC86] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.
- [SHB04] Cyrill Stachniss, Dirk Hähnel, and Wolfram Burgard. Exploration with active loop-closing for FastSLAM. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, September 28 - October 2, 2004*, volume 2, pages 1505–1510, Sendai, Japan, September/October 2004. IEEE.
- [SK08] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2008.
- [SNH03] Hartmut Surmann, Andreas Nüchter, and Joachim Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3):181–198, 2003.
- [SNLH09] Jochen Sprickerhof, Andreas Nüchter, Kai Lingemann, and Joachim Hertzberg. An explicit loop closing technique for 6d slam. In *European Conference on Mobile Robots, ECMR 2009*, pages 229–234, 2009.
- [SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.



- [Sup08] Michael Suppa. *Autonomous Robot Work Cell Exploration Using Multisensory Eye-in-hand Systems*. PhD thesis, Universität Hannover, 2008.
- [T<sup>+</sup>02] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millennium*, pages 1–35, 2002.
- [TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, MA, 2005.
- [TLK<sup>+</sup>04] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.
- [TMM<sup>+</sup>06] Benjamín Tovar, Lourdes Muñoz, Rafael Murrieta, Moisés Alencastre, Raúl Monroy, and Seth Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.
- [VAC13] Joan Vallvé and Juan Andrade-Cetto. Mobile robot exploration with potential information fields. In *European Conference on Mobile Robots, ECMR 2013*, pages 222–227. IEEE, 2013.
- [WHB<sup>+</sup>10] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010. Software available at <http://octomap.sf.net/>.
- [WSB08] Kai M. Wurm, Cyrill Stachniss, and Wolfram Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22–26, 2008, Acropolis Convention Center, Nice, France*, pages 1160–1165, Acropolis Convention Center, Nice, France, September 2008. IEEE.
- [WSBC10] Jochen Welle, Dirk Schulz, Thomas Bachran, and Armin B. Cremers. Optimization techniques for laser-based 3d particle filter slam. In *2010 IEEE International Conference on Robotics and Automation, ICRA 2010, Anchorage, Alaska, USA, 3–7 May 2010*, pages 3525–3530, Anchorage, Alaska, USA, May 2010. IEEE.
- [Yam97] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997, CIRA '97, Proceedings*, pages 146–151. IEEE, 1997.

- [Zha00] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

# Statutory Declaration

I, Thomas Wohlfahrt, herewith declare that I have completed the present thesis independently making use only of the specified literature and aids. Sentences or parts of sentences quoted literally are marked as quotations; identification of other references with regard to the statement and scope of the work is quoted. I further declare that this work has not been submitted for credit elsewhere.

10.5.2015, Minden  
(place, date)

T. Wohlfahrt  
(signature)